

# Projet N°227 : Caravel

Rapport

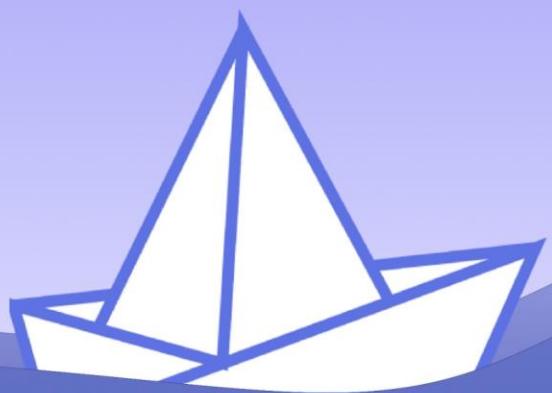
Travail de Bachelor

**Professeurs :** David GRUNENWALD, Alexander WOHLFAHRT

**Expert :** Julien M'POY

Mendes Reis Steve

29 juillet 2021



## Table des matières

Résumé . . . . .	8
Abstract . . . . .	8
Remerciements . . . . .	9
<b>1 Introduction</b>	<b>10</b>
<b>2 Cahier des charges</b>	<b>11</b>
2.1 Buts visés . . . . .	11
2.1.1 Priorisation des tâches . . . . .	12
<b>3 Analyse</b>	<b>14</b>
3.1 Problématique . . . . .	14
3.1.1 Pour les professeurs . . . . .	14
3.1.2 Pour les élèves . . . . .	14
3.2 Etat de l'art . . . . .	15
3.2.1 Applications testées . . . . .	15
3.2.2 Synthèse . . . . .	22
3.2.3 Positionnement de Caravel . . . . .	22
3.3 Conception . . . . .	23
3.3.1 Rôles . . . . .	23
3.3.2 <i>Use cases</i> . . . . .	24
3.3.3 Gestion des filtres . . . . .	29
3.3.4 Système de réactions . . . . .	29
3.3.5 Système de notifications . . . . .	30
3.3.6 Gestion de la charge de travail . . . . .	31
3.3.7 Modélisation de la base de données . . . . .	33
3.3.8 Système d'authentification . . . . .	35
3.4 Définitions des routes . . . . .	35
3.5 Stratégie & conception de test . . . . .	35
3.5.1 Attentes de la qualité du produit . . . . .	35
3.5.2 Objectifs de tests . . . . .	35
3.5.3 Périmètre de tests . . . . .	36
3.5.4 Gestion des risques . . . . .	36
3.6 Maquettes . . . . .	37
3.6.1 Page de login . . . . .	37
3.6.2 Page des tâches . . . . .	38
3.6.3 Page d'affichage d'une tâche . . . . .	39

3.6.4	Page vue mensuelle . . . . .	40
3.6.5	Page chronologique . . . . .	41
3.6.6	Page de statistiques . . . . .	42
3.6.7	Maquette interactive . . . . .	43
3.7	Planning . . . . .	43
3.8	Méthodologie de travail . . . . .	44
<b>4</b>	<b>Implémentation</b>	<b>45</b>
4.1	DevOps CI/CD . . . . .	45
4.1.1	Intégration continue . . . . .	45
4.1.2	Livraison continue . . . . .	46
4.1.3	Environnement de production . . . . .	49
4.2	Authentification . . . . .	49
4.2.1	Local Storage vs Cookies . . . . .	50
4.2.2	Sanctum vs Passport . . . . .	50
4.2.3	Authentification LDAP . . . . .	52
4.3	Intégration des routes . . . . .	55
4.3.1	Routes backend . . . . .	55
4.3.2	Routes frontend . . . . .	55
4.4	Gestion des réactions . . . . .	55
4.4.1	Gestion de l'identifiant . . . . .	55
4.4.2	Représentation des réactions . . . . .	56
4.4.3	Affichage des réactions . . . . .	57
4.4.4	Incitation aux réactions . . . . .	57
4.5	Système de notification . . . . .	57
4.5.1	Déclenchement d'une notification . . . . .	58
4.5.2	Configuration de FCM . . . . .	60
4.5.3	Récupération des notifications depuis le frontend . . . . .	60
4.6	Frontend . . . . .	61
4.6.1	Configuration Vue.js . . . . .	61
4.6.2	Vuex . . . . .	63
4.6.3	Gestion des erreurs Axios . . . . .	70
4.6.4	Vue router . . . . .	71
4.6.5	PWA . . . . .	72
4.6.6	Composants . . . . .	73
4.6.7	Editeur markdown . . . . .	75
4.7	Backend . . . . .	76
4.7.1	Choix de la base de données . . . . .	76

4.7.2	Middleware . . . . .	76
4.7.3	Policies . . . . .	78
4.7.4	Validation des requêtes . . . . .	78
4.7.5	Moteur de recherche . . . . .	78
<b>5</b>	<b>Tests et validation</b>	<b>80</b>
5.1	Test unitaire . . . . .	81
5.2	Qualité du code . . . . .	81
5.3	Test d'utilisabilité . . . . .	81
5.3.1	Scénario de test . . . . .	81
5.3.2	Résultats . . . . .	82
<b>6</b>	<b>Améliorations</b>	<b>83</b>
6.1	Système de notification . . . . .	83
6.1.1	Système d'abonnement . . . . .	83
6.2	Gestion des filtres en frontend . . . . .	83
6.3	Editeur markdown . . . . .	83
6.3.1	Ajout de balises spécifiques . . . . .	84
6.4	Ajout des groupes de manière automatique . . . . .	84
6.5	Ajout de paramètres de synchronisation avec LDAP . . . . .	84
6.6	Gestion des suppressions . . . . .	84
<b>7</b>	<b>Conclusion</b>	<b>85</b>
<b>8</b>	<b>Glossaire</b>	<b>86</b>
<b>9</b>	<b>Annexes</b>	<b>87</b>
9.1	Guide d'installation . . . . .	87
9.2	Planning (au format GanttProject et PNG) . . . . .	87
9.3	Journal de travail . . . . .	87
<b>10</b>	<b>Références</b>	<b>88</b>

## Table des figures

1	Google Classroom : liste des types de contenu . . . . .	16
2	Google Classroom : gestion de liste d'élèves . . . . .	16
3	Google Classroom : permet de faire des annonces . . . . .	17
4	Google Classroom : vue limité à la semaine . . . . .	17
5	MyHomework : ajout d'un devoir . . . . .	18
6	MyHomework : vue centrale simple et efficace . . . . .	18
7	MyHomework : vue calendrier (mensuelle) . . . . .	19
8	MyHomework : vue semaine . . . . .	19
9	MyStudyLife : gestion des tâches . . . . .	20
10	MyStudyLife : gestion des différents sujets . . . . .	21
11	MyStudyLife : gestion des cours . . . . .	21
12	<i>Use case</i> : gestion des tâches . . . . .	24
13	<i>Use case</i> : gestion des sujets . . . . .	25
14	<i>Use case</i> : gestion des fils de discussion . . . . .	26
15	<i>Use case</i> : gestion des notifications . . . . .	27
16	<i>Use case</i> : gestion des filtres . . . . .	28
17	<i>Use case</i> : gestion du login . . . . .	28
18	Maquette des paramètres de notifications . . . . .	31
19	Modélisation de la base de données . . . . .	34
20	Maquette : page de login . . . . .	38
21	Maquette : page liste des tâches . . . . .	39
22	Maquette : page affichage d'une tâche . . . . .	40
23	Maquette : page vue mensuelle . . . . .	41
24	Maquette : page vue timeline . . . . .	42
25	Maquette : page de statistiques . . . . .	43
26	Méthodologie : GitFlow workflow feature branches . . . . .	44
27	Authentification : schéma d'interaction . . . . .	50
28	Laravel Sanctum Explained : SPA Authentication (voir 6 ) . . . . .	51
29	Notification : schéma global du système de notifications . . . . .	57
30	Frontend : configuration Vue.js . . . . .	62
31	Vuex : one-way data flow, voir référence (11) . . . . .	63
32	Vuex : détails de fonctionnement, voir référence (11) . . . . .	64
33	Vuex : liste des modules . . . . .	65
34	Charge de travail : gestion des projets traversants . . . . .	67
35	PWA : score lighthouse . . . . .	72
36	Vue.js : composant sélection du type . . . . .	73

---

37	Vue.js : composant sélection du sujet . . . . .	73
38	Vue.js : modale de création d'une fenêtre . . . . .	74
39	Laravel : schéma de fonctionnement d'un middleware . . . . .	76
40	Laravel : structure des fichiers du moteur de recherche . . . . .	79

## Liste des codes

1	DevOps : pipeline de test Laravel SQLite . . . . .	45
2	Serveur : installation des dépendances de base . . . . .	46
3	Serveur : création de la db et d'un user particulier . . . . .	47
4	Serveur : configuration Nginx . . . . .	47
5	Authentification : fonction de login . . . . .	51
6	LDAP : création d'un annuaire avec docker . . . . .	52
7	LDAP : synchronisation des champs . . . . .	52
8	LDAP : gestion de paramètre spécifique . . . . .	53
9	LDAP : gestion du login et du fallback . . . . .	54
10	Gestion de la serialisation des réactions . . . . .	56
11	Notifications : canaux de diffusion . . . . .	59
12	Notification : comportement des canaux . . . . .	59
13	Notification : envoi asynchrone . . . . .	60
14	Notification : enregistrement du token FCM . . . . .	61
15	Notification : call asynchrone des messages . . . . .	61
16	Vuex : chargement des dépendances . . . . .	65
17	Vuex : module générique . . . . .	68
18	Vuex : gestion du chargement . . . . .	69
19	Axios : gestion des erreurs globales avec les interceptors . . . . .	70
20	Vue.js : protection des routes . . . . .	71
21	Vue.js : exemple de lazy loading . . . . .	72
22	Vue.js : exemple d'utilisation du composant paginate . . . . .	74
23	Vue.js : code requis sans l'utilisation de la pagination . . . . .	75
24	Laravel : vérification des droits de groupe . . . . .	77
25	Laravel : exemple de contrôleur avec une classe de validation . . . . .	78
26	Laravel : exemple d'utilisation du moteur dans un contrôleur . . . . .	79
27	Laravel : exemple de filtre . . . . .	80

## Résumé

Caravel est un projet initié par des trois étudiants lors du cours de “Développement web” mené à la He-Arc, il a été ensuite promu en travail de Bachelor afin d’obtenir un produit réellement utilisable au sein de la he-arc.

Le but de cette application est de permettre la collaboration étroite entre professeurs et élèves concernant les devoirs, tests, examens ou projets. Elle répond à une problématique souvent soulevée par les élèves qui est la mauvaise répartition de la charge de travail dans les différentes semaines. Caravel permet donc une meilleure visibilité de la charge de travail des différentes semaines afin que les professeurs puissent placer au mieux leur devoirs.

## Abstract

Caravel is a project initiated by three students during the “Web Development” course at He-Arc, it was then promoted to a Bachelor work in order to obtain a product really usable within He-Arc.

The goal of this application is to allow close collaboration between teachers and students concerning assignments, tests, exams or projects. It responds to a problem often raised by students which is the poor distribution of the workload in the different weeks. Caravel allows a better visibility of the workload of the different weeks so that the teachers can place their homework in the best possible way.

---

Projet N°227 : Caravel

He-Arc

29 juillet 2021

## **Remerciements**

Merci à Luca Truscello<sup>1</sup> étudiant à L'HEPIA de Genève qui a produit le logo de Caravel.

Merci à MM. Grunenwald, Wohlfahrt et M'Poy, pour le suivi et les conseils avisés.

Et un merci particulier à Joris Monnet et Maxime Welcklen sans qui ce projet n'aurait pu exister.

---

<sup>1</sup>truscello.luca@gmail.com

## 1 Introduction

Le projet Caravel fait suite à un travail réalisé par trois étudiants dans le cadre du cours de *Développement Web*. Lors de ce projet les trois étudiants ont réalisé une plateforme web permettant aux élèves d'inscrire de manière collaborative les différents tâches (devoirs/CP/Projet) à faire pour leur classe.

Le but de ce projet est donc de pousser Caravel plus loin en y ajoutant de nouvelles fonctionnalités pour le rendre utilisable par la He-Arc.

Le projet s'articule autours des objectifs suivants :

- Placer au mieux les devoirs pour lisser la charge et pouvoir voir si la charge de travail est correcte.
- Réunir en un seul endroit toutes les informations relatives au travail qui doit être effectué par les étudiants et ainsi éviter l'utilisation de différents canaux.
- Aider les élèves à mieux s'organiser en ayant une place qui réuni toutes les informations nécessaires à l'exécution de leur travail.
- Proposer une plateforme de collaboration entre étudiants et professeurs sur des tâches via des échanges questions/réponses.

Dans la suite du document nous allons voir en détails les objectifs de Caravel. Puis à partir de ces objectifs, une analyse des différents problématiques sera proposée. Une fois l'analyse exposée, la place sera laissée à la conception dans laquelle nous verrons de manière abstraite comment résoudre les différentes problématiques relatives au projet. Ces aspects éclaircis, nous pourrons passer à l'implémentation de la solution. Finalement nous verrons certains points d'amélioration au sein de la solution proposée, pour finir par une conclusion dans laquelle nous reviendrons sur les objectifs introduits préalablement.

## 2 Cahier des charges

Cette section décrit le cadre du projet ainsi que ses buts visés à la fin du développement

### 2.1 Buts visés

Avec Caravel l'idée est que les devoirs soient gérés par les membres d'une classe, chaque membre de la classe a donc la possibilité de renseigner un devoir sur la plateforme. Si celui-ci manque de précision, le professeur ou un élève peut y apporter une modification en tout temps afin d'éclaircir son contenu. Dans cette optique autant les professeurs que les élèves peuvent être une source de devoirs ce qui change la dynamique par rapport aux applications habituelles.

Un des buts de Caravel est de réunir tous les devoirs en un seul endroit, avec l'apparition des différents outils comme Moodle, Teams, l'intranet ou encore les dossiers partagés, il est parfois très difficile de savoir où chercher l'information concernant un devoir. L'idée est donc de permettre aux membres d'une classe de réunir les informations pertinentes en un seul endroit afin de gagner du temps. La contribution collaborative permet de faciliter la transition à l'utilisation de Caravel, si par exemple un professeur continue de distribuer ses exercices sous format PDF par des canaux différents, il suffit qu'un seul élève soit au courant de l'information pour la transmettre aux autres en les ajoutant sur Caravel.

De plus Caravel pourra permettre d'ouvrir des fils de discussion au sein d'une tâche afin de demander de l'aide ou des éclaircissement sur une tâche, l'idée est de permettre aux élèves de s'entraider et de partager en un seul lieu les différentes informations (questions/réponses).

Enfin, l'outil devra permettre d'avoir une vue de la charge de travail d'une classe afin de placer au mieux les prochains CP/devoirs, il sera également possible pour un élève de réagir sur les différentes tâches pour alerter les professeurs sur un devoir qui serait inadapté (temps de travail, complexité, manque d'informations, etc...)

Succinctement les buts sont les suivants :

- Placer au mieux les devoirs et CP pour lisser la charge et savoir si la charge de travail est correcte
- Réunir en un seul endroit toutes les informations relatives au travail qui doit être effectué par l'étudiant et ainsi éviter l'utilisation de différents canaux
- Aider les élèves à mieux s'organiser en ayant une place qui réuni toutes les informations nécessaires à l'exécution de leur travail
- Proposer une plateforme de collaboration entre étudiants et professeurs sur des tâches via des échanges questions/réponses

### **2.1.1 Priorisation des tâches**

Une priorisation des tâches a été effectuée enfin de déterminer les éléments importants du projet. Cette priorisation est basée sur la méthode de MoSCoW (voir 1).

#### **2.1.1.1 Must have**

- Traduction en français
  - Prévoir la possibilité d'ajouter d'autres langues facilement (localisation)
- Filtres sur les tâches dans les différentes vues
  - Filtres par titre, par sujet, par auteur
- Rôles étudiant/professeur
- Ajout d'une vue en mois style Outlook
  - Permet de naviguer sur d'anciennes semaines
- Séparation front/back end
  - Passage en Vue.js pour le frontend et Laravel pour le backend
- Analyse & intégration des feedbacks reçus pendant le semestre de printemps
- Réactions aux différentes tâches
  - Pouvoir réagir une tâche (trop long, trop complexe, etc...)
- Ajout de notion de crédits ECTS sur les sujets

#### **2.1.1.2 Should have**

- Système de notifications
  - Par exemple, une tâche a été ajoutée ou modifiée, réponse à un commentaire
- Authentification interne (LDAP)
- Système de sujets (fils) dans les commentaires au niveau des tâches
  - Possibilité d'éditer les commentaires
- Ajout de la représentation de la charge de travail

### 2.1.1.3 Could have

- PWA
  - Possibilité de notification push
- Possibilité pour un élève de mettre une tâche comme terminée pour lui uniquement
- Partage des tâches privées avec certains membres
- Enrichir l'éditeur de texte
  - ajouter des mentions type @membre, #idTache
- Gestion des paramètres de notifications
- Onglet "Mes tâches" pour qu'un utilisateur puisse retrouver facilement ses tâches créées
- Contrôle édition d'une tâche simultanée (simpliste)
  - Empêcher la soumissions d'un formulaire si la date de modification a changé entre temps
- Approvisionnement automatique des groupes de classes (étudier la faisabilité)
  - Préremplis avec certains professeurs comme membres
- Vue Semestrielle (type Gantt)
- Une vue récap/statistique globale pour un groupe

### 2.1.1.4 Won't have

- Pas de gestion d'édition collaborative simultanée (web socket, style google docs)
- Pas de gestion des tâches transversales (sur plusieurs groupes)

### 3 Analyse

La section suivante décrit la partie d'analyse et de conception qui a été faite en amont pendant le dernier semestre de troisième année.

#### 3.1 Problématique

La problématique est divisée en deux parties, une partie concernant les professeurs et une autre les élèves.

##### 3.1.1 Pour les professeurs

**3.1.1.1 Problème 1** Placer au mieux les devoirs et CP pour lisser la charge et savoir si la charge de travail est correcte.

**3.1.1.1.1 Solution** Elle repose sur deux propositions, dans un premier temps fournir une vue qui permette au mieux de placer un CP ou un devoir (tout en discutant avec les élèves). Dans un deuxième temps il sera possible pour un élève de réagir sur un devoir à l'aide de réactions qui permettront d'évaluer un devoir (trop long/trop complexe/etc...). Les professeurs pourront alors voir si un devoir a occasionné beaucoup de réactions et donc s'il était adapté ou non. Ces réactions pourront entraîner par la suite des discussions avec les élèves pour améliorer la tâche et à fortiori la participation des élèves.

**3.1.1.2 Problème 2** Les professeurs distribuent souvent des consignes de manière orale ou alors sur des supports spécifiques, avec les différents outils disponibles, mettre les informations sur toutes les différentes plateformes peut être éreintant.

**3.1.1.2.1 Solution** Possibilité de déléguer cette tâche aux élèves (ex. le devoir peut être donné de manière orale et être introduit sur Caravel par un élève), étant plus nombreux il est plus facile pour les élèves de centraliser les informations qui leur sont nécessaires pour un devoir plutôt que de laisser cette action à une seule et unique personne.

##### 3.1.2 Pour les élèves

**3.1.2.1 Problème** Les devoirs sont notés et épargnés sur plusieurs supports (physique ou digital) parmi les élèves, nous nous retrouvons souvent avec un élève qui détient une partie de l'information

et non toute l'information. Les élèves ont donc du mal à visualiser tous les devoirs à faire. Il est alors compliqué de prévoir sa charge de travail avec des informations incomplètes.

**3.1.2.1.1 Solution** Apporter une vue centralisée dans laquelle il est facile de visualiser les tâches à faire, l'accès collaboratif permet de réunir les fragments d'informations détenus par chaque élève en un seul endroit afin d'obtenir une information complète.

### 3.2 Etat de l'art

Cette section décrit l'état des applications dans le domaine de gestion des tâches liés au monde étudiantin, elle décrit notamment quelques tests effectués et fonctionnalités intéressantes sur certaines de ces applications et ultimement le positionnement de Caravel par rapport à l'état de l'art.

#### 3.2.1 Applications testées

- Google Classroom<sup>2</sup>
- MyHomework<sup>3</sup>
- MyStudyLife<sup>4</sup>

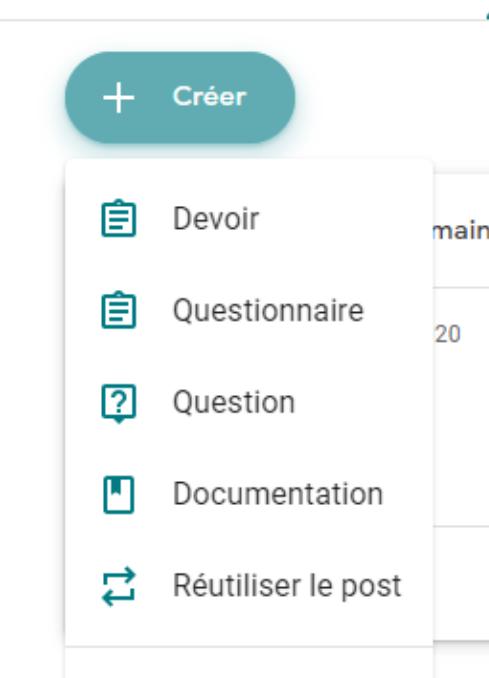
**3.2.1.1 Google Classroom** L'outil Google Classroom est très axé cours, les professeurs créent leur cours et les élèves suivent les cours qui leur sont nécessaires (principe universitaire où les élèves peuvent choisir leurs cours). L'idée est que les ajouts tels que les devoirs soient introduits uniquement par le professeur. La notion de classe à proprement parler n'existe pas.

**3.2.1.1.1 Type de publication** Il est possible de publier différents types de contenu dans Classroom.

<sup>2</sup><https://classroom.google.com/>

<sup>3</sup><https://myhomeworkapp.com/home>

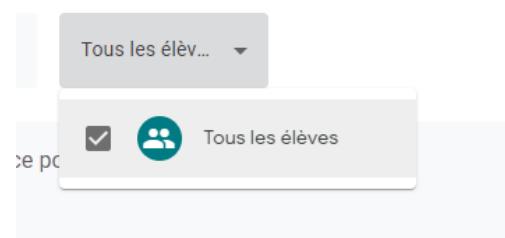
<sup>4</sup><https://app.mystudylife.com/>



**FIG. 1 :** Google Classroom : liste des types de contenu

Comme le montre la figure 1, celle-ci ne se limite pas seulement à des devoirs mais aussi à des questions.

**3.2.1.1.2 Gestion des accès** Les accès peuvent être gérés finement dans Google Classroom.



**FIG. 2 :** Google Classroom : gestion de liste d'élèves

Ainsi il est possible de créer des groupes d'élèves afin de faire des partages spécifiques, comme le montre la figure 2.

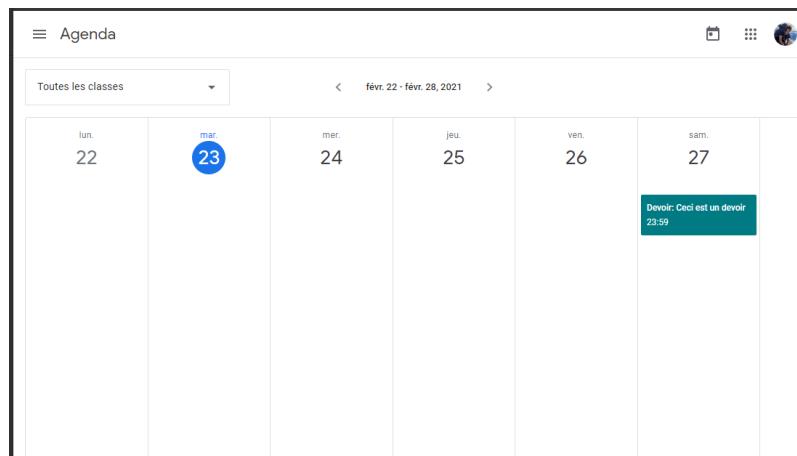
**3.2.1.1.3 Annonces** Des annonces peuvent être émises par tous au sein de la classe.



**FIG. 3 :** Google Classroom : permet de faire des annonces

Comme le montre la figure 3, ces annonces apparaissent dans la vue centrale de l'application, ce qui peut être pratique pour faire des rappels.

**3.2.1.1.4 Vue calendrier** La vue calendrier disponible pour les différentes tâches est limitée à la semaine.



**FIG. 4 :** Google Classroom : vue limité à la semaine

Comme le montre la figure 4, cela donne une bonne idée de la charge de la semaine, il est cependant dommage de ne pas trouver une vue mensuelle.

**3.2.1.1.5 Conclusion** Fonctionnalités intéressantes :

- Des annonces peuvent être faites pour la classe
- Il y a une notion de groupes, les devoirs peuvent être distribués à toute la classe ou alors à un groupe plus restreint

- Les devoirs sont synchronisés directement avec l'agenda Google

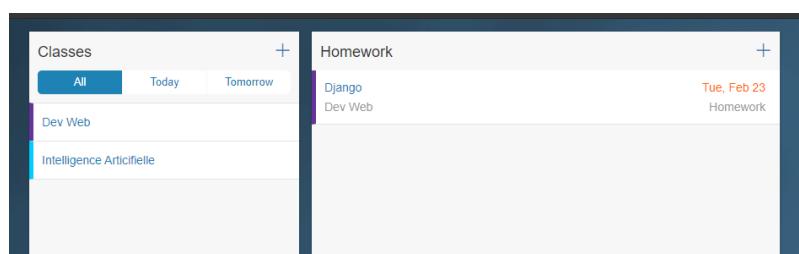
**3.2.1.2 MyHomework** MyHomework est une WebApp qui permet la gestion des devoirs personnels.

**3.2.1.2.1 Ajout de devoir** L'interface d'ajout d'un devoir est très lourde.

**FIG. 5 :** MyHomework : ajout d'un devoir

Il y a beaucoup d'options qui, la plupart du temps, ne sont pas nécessaires, voir la figure 5.

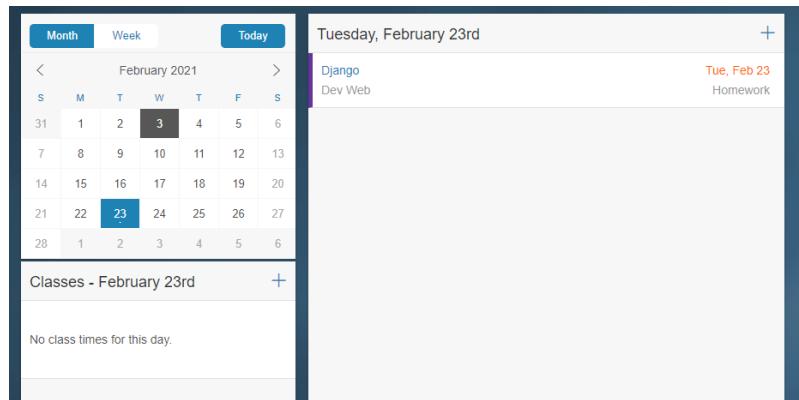
**3.2.1.2.2 Vue centrale simple** La vue centrale de l'application est claire et efficace.



**FIG. 6 :** MyHomework : vue centrale simple et efficace

Elle permet de voir simplement les tâches qui doivent être effectuées, comme nous pouvons le voir dans la figure 6.

**3.2.1.2.3 Vue mois** La vue mois est très basique, elle ne permet pas de voir directement les tâches à effectuer.



**FIG. 7 :** MyHomework : vue calendrier (mensuelle)

Il faut cliquer sur le jour pour avoir des détails sur les tâches, comme nous pouvons le voir sur la figure 7.

**3.2.1.2.4 Vue semaine** La vue semaine permet de clairement voir les tâches à effectuer pour la semaine en cours.



**FIG. 8 :** MyHomework : vue semaine

Elle contient assez de détails pour ne pas être obligé d'ouvrir la tâche, comme le montre la figure 8, ce qui est très pratique.

**3.2.1.2.5 Conclusion** L'application rend le service minimum (ajout de devoirs) mais ne permet pas la collaboration, il y a beaucoup d'options qui peuvent perdre l'utilisateur.

Fonctionnalités intéressantes :

- Affichage en semaine concise
- Possibilité de rajouter des rappels
- Ajout d'une notion "terminé"

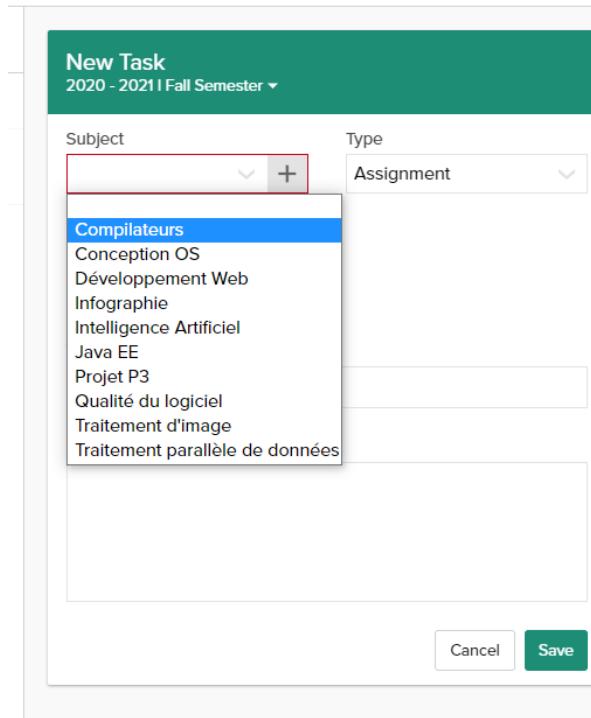
**3.2.1.3 MyStudyLife** MyStudyLife est une WebApp (disponible aussi sur mobile) qui permet la gestion des horaires de cours ainsi que la gestion des tâches à effectuer.

**3.2.1.3.1 Vue centrale** La vue centrale de l'application permet de visualiser clairement les différentes tâches à effectuer.

**FIG. 9 :** MyStudyLife : gestion des tâches

Comme le montre la figure 9, celle-ci est très simple, de plus il est possible de directement marquer les tâches comme terminées.

**3.2.1.3.2 Gestion des sujets** La gestion des sujets a l'avantage d'être très simple.



**FIG. 10 :** MyStudyLife : gestion des différents sujets

Il est possible de facilement ajouter un nouveau sujet depuis la création d'une tâche, comme le montre la figure 10.

**3.2.1.3.3 Gestion des cours** Il est possible de gérer son horaire de cours de manière très fine, par exemple il est possible d'inscrire un cours avec des rotations (une semaine sur deux) ou uniquement sur un semestre.

Classes	New Class	Holidays	New Holiday
0 Compilateurs: 3253.2 08:15 - 11:35 mercredi	FRT	Fall Break avr. 5 - 9 2021	
1 Conception OS: 3253.1 10:00 - 11:35 lundi	CCO	Red Week mai 10 - 14 2021	
Conception OS: 3253.1 12:30 - 14:50 vendredi	CCO		
Développement Web: 3255.2 13:15 - 14:50 lundi	DGR		
Infographie: 3252.1 15:05 - 17:25 jeudi	BLC		
Intelligence Artificiel: 3256.1 08:15 - 10:45 mercredi	STC		
Intelligence Artificiel: 3256.1 08:15 - 09:45 jeudi	STC		
Java EE: 3257.2 10:00 - 12:25 lundi	CHE_X		

**FIG. 11 :** MyStudyLife : gestion des cours

Comme il est possible de voir sur la figure 11. Ceci permet d'automatiquement pré-remplir certains champs, par exemple lors d'un ajout d'une tâche si un cours est en cours alors la tâche sera pré-remplie avec ce cours.

### **3.2.1.3.4 Conclusion** Fonctionnalités intéressantes :

- Gestion des horaires de cours très flexible
- Permet de lier des tâches à un cours et d'effectuer des rappels avant le début du cours
- Permet de gérer les vacances
- Permet de gérer les cours sur différents semestres
- Les notifications push disponibles avec l'application sont très appréciables
- Possède une version web et mobile

### **3.2.2 Synthèse**

Le marché est partagé en deux catégories : les applications de gestion de devoirs dans lesquelles c'est l'étudiant qui entre les devoirs et l'autre catégorie où ce sont les professeurs qui ajoutent les devoirs.

Pour la première catégorie, il existe actuellement beaucoup d'applications (surtout mobile) qui permettent à un étudiant de gérer ses devoirs mais celles-ci ne permettent pas la collaboration entre étudiants, en outre il n'est pas possible de partager les devoirs avec quelqu'un d'autre.

Pour la seconde catégorie, il existe quelques applications qui permettent à un professeur d'entrer des devoirs pour un groupe d'élèves, ceux-ci peuvent alors tous consulter les devoirs. Le problème repose sur le fait que seul le professeur peut entrer les devoirs, cela limite la marge de manœuvre des élèves ainsi que leur implication. Dans ces applications un professeur est souvent responsable de sa matière et ne peut donc pas forcément rajouter des devoirs dans une autre matière.

### **3.2.3 Positionnement de Caravel**

Dans Caravel l'idée est de se positionner entre les deux mondes, les devoirs seront gérés par les membres d'une classe, chaque membre de la classe a donc la possibilité de renseigner un devoir sur la plateforme. Si celui-ci manque de précision, le professeur ou un élève peut y apporter une modification en tout temps afin d'éclaircir son contenu. Dans cette optique autant les profs que les élèves peuvent être une source de devoirs ce qui change la dynamique des élèves par rapport aux applications habituelles.

Un des souhaits de Caravel est aussi de réunir tous les devoirs en un seul endroit. Avec l'apparition des différents outils comme Moodle, Teams, l'intranet ou encore les dossiers partagés, il est parfois

très difficile de savoir où chercher l'information concernant un devoir. L'idée est donc de permettre aux membres de la classe de réunir les informations pertinentes en un seul endroit afin de gagner du temps.

De plus Caravel pourra permettre d'ouvrir des fils de discussion au sein d'une tâche afin de demander de l'aide ou des éclaircissement sur un détail spécifique, l'idée est de permettre aux élèves de s'entraider sur une question et de partager en un seul lieu les différentes informations sur un devoir, évidemment les professeurs peuvent aussi répondre aux différents fils de discussion.

Enfin, l'outil permet aux élèves de toujours être à jour quant aux tâches à faire, en effet il arrive souvent qu'un élève oublie de noter un devoir ou de détailler certains éléments, Caravel permet de réunir les connaissances des différents élèves et des professeurs pour obtenir une base solide d'informations.

### 3.3 Conception

Cette partie décrit les problématiques ainsi que les solutions qui ont été trouvées.

#### 3.3.1 Rôles

Cette partie explicite les différents rôles disponibles au sein de caravel ainsi que les actions possibles.

**3.3.1.1 Rôles fonctionnels** Il y a un seul rôle fonctionnel qui est celui d'administrateur du groupe, en général il s'agit du créateur de ce dernier mais ce droit peut être transmis.

**3.3.1.1.1 Administrateur du groupe** Permet de gérer les paramètres du groupe (suppression) ainsi que de gérer les différents membres du groupes (suppression d'un membre).

**3.3.1.2 Rôles sémantiques** Les rôles suivants n'auront pas de droits particuliers, le but de l'application étant de permettre la collaboration directe entre eux, cependant les actions des professeurs seront mises en avant, typiquement dans les fils de discussions. De plus les professeurs seront admis d'office dans les groupes dit de "classe" lors d'une demande d'adhésion.

Les deux rôles seront donc les suivants :

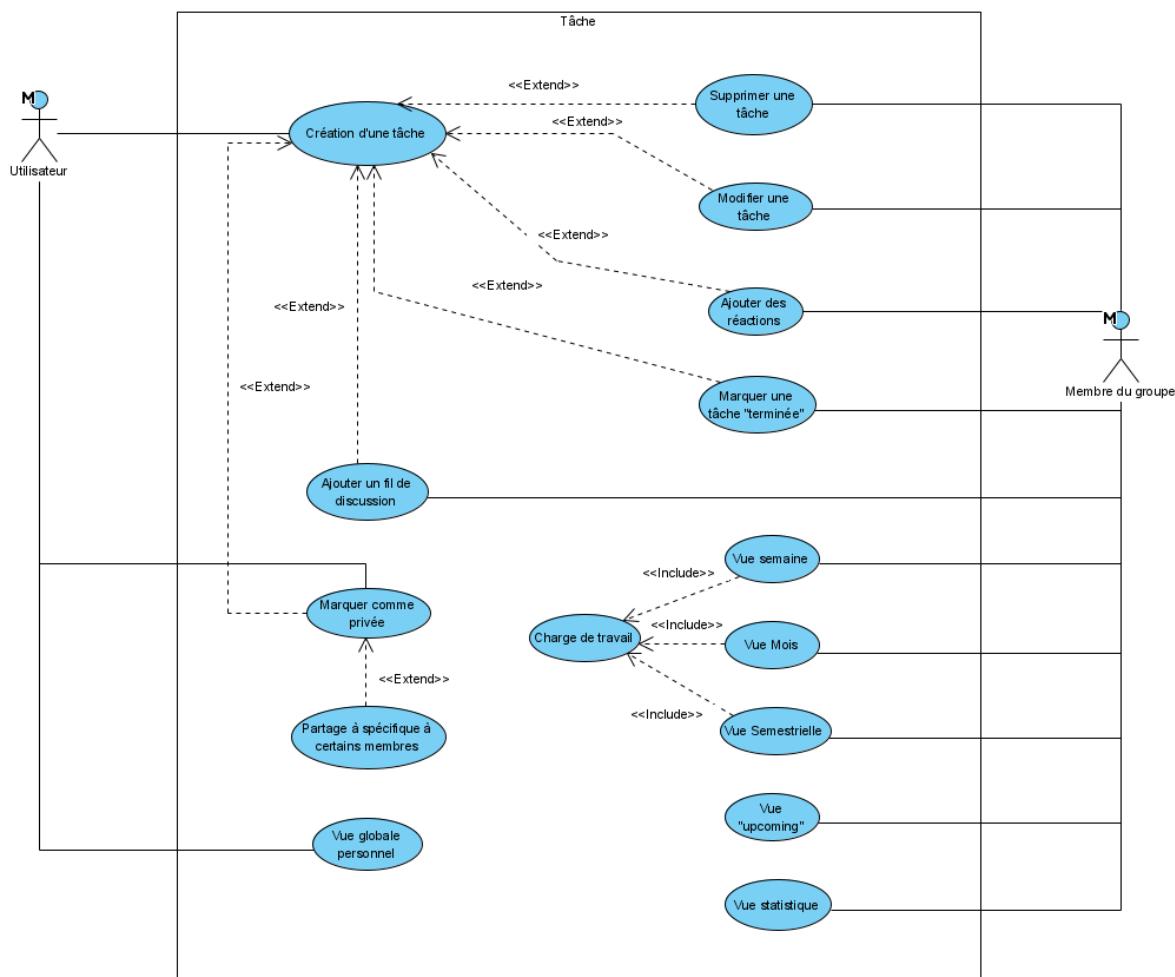
- Professeur
- Elève

Il a été décidé de ne pas appliquer de droit spécifique pour l'un ou l'autre des rôles car un historique des actions de chaque utilisateur sera mis en place et donc il possible en tout temps de trouver qui a effectué la moindre modification sur une tâche, tout utilisateur est libre de modifier une tâche même s'il en est pas l'auteur. Ceci afin d'encourager la collaboration.

### 3.3.2 Use cases

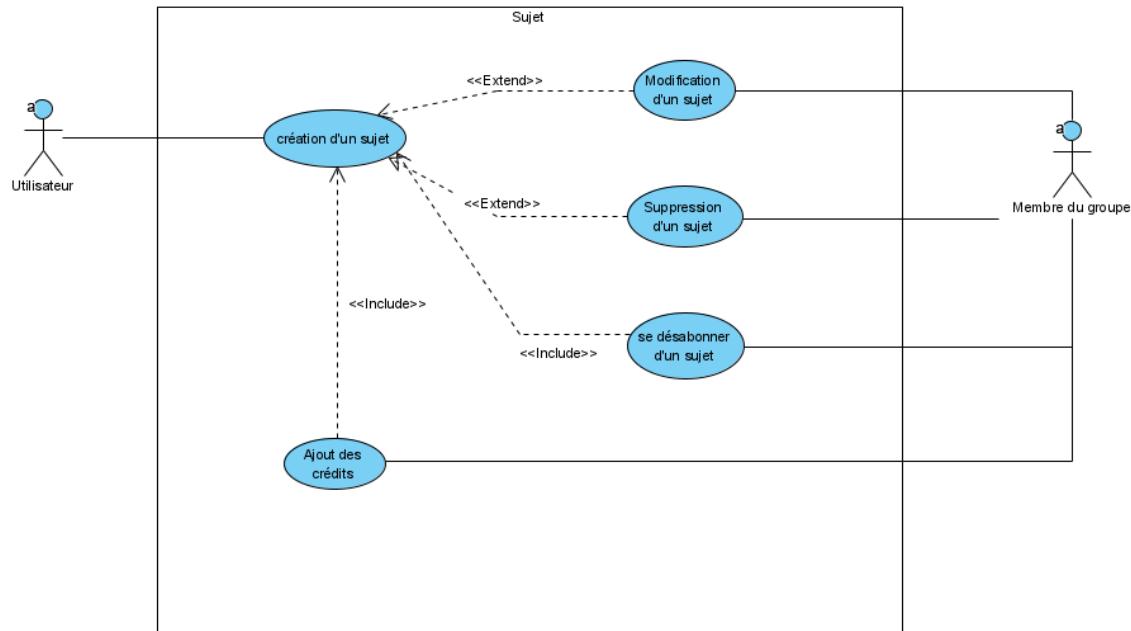
Pour la bonne compréhension des schémas qui vont suivre, il tenir compte du fait qu'un **Utilisateur** est un **Membre du groupe**. De plus l'utilisateur est aussi considéré comme un **Auteur**.

#### 3.3.2.1 Tâche Use case concernant les différentes actions possibles sur les tâches.



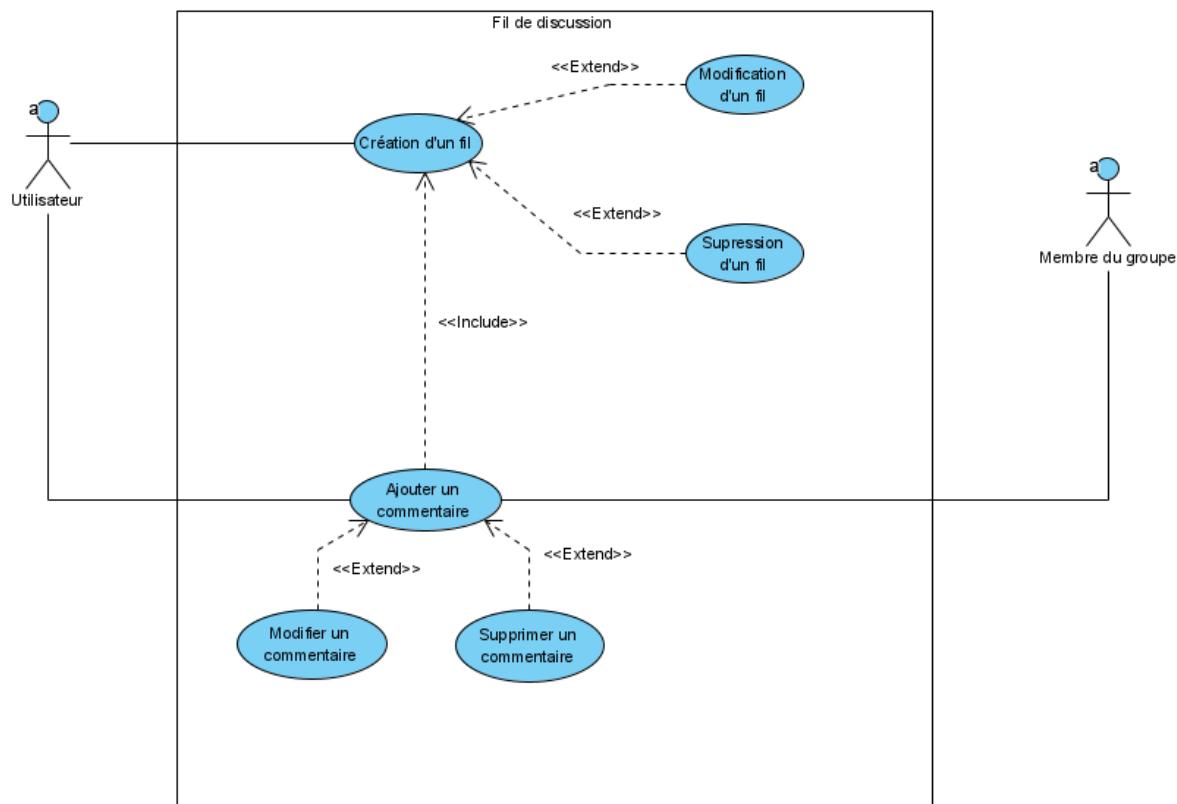
**FIG. 12 :** Use case : gestion des tâches

**3.3.2.2 Sujet** Use case concernant les différentes actions possibles sur les sujets.



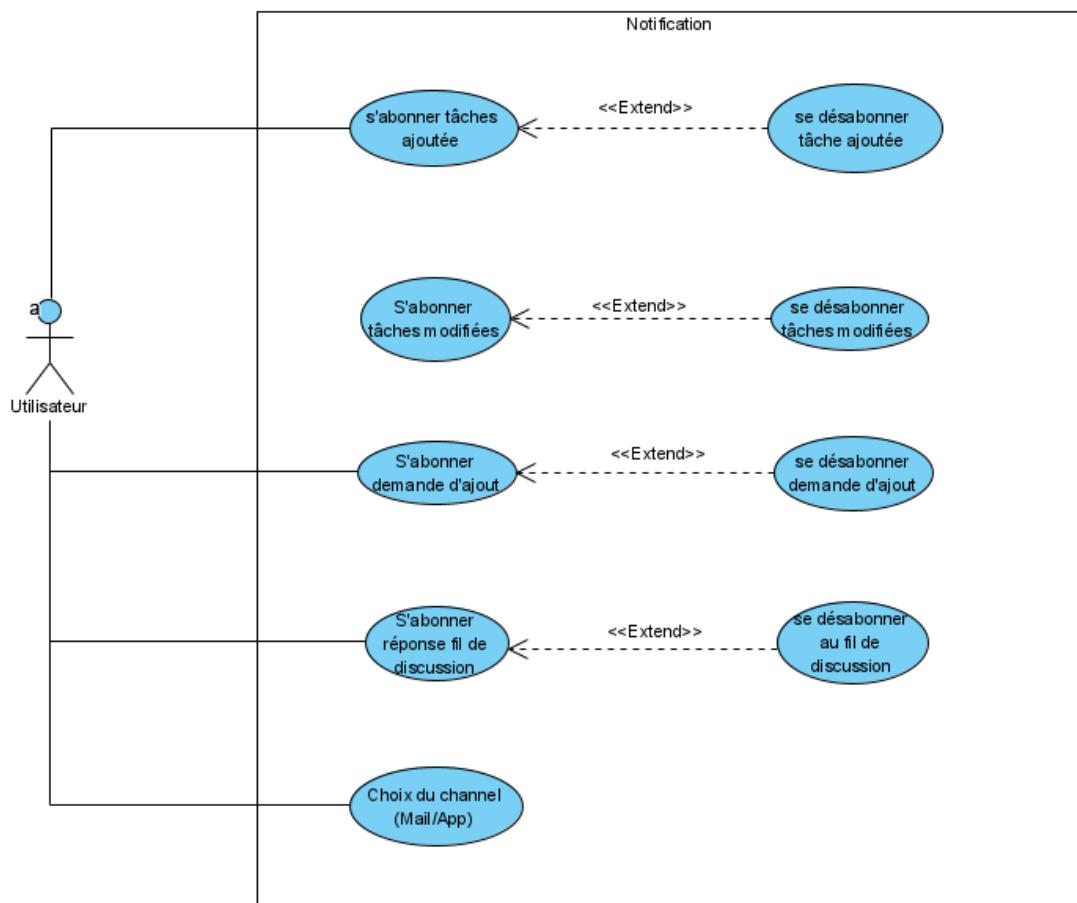
**FIG. 13 :** Use case : gestion des sujets

**3.3.2.3 Fil de discussion** Use case concernant les différentes actions possibles sur les fils de discussion.



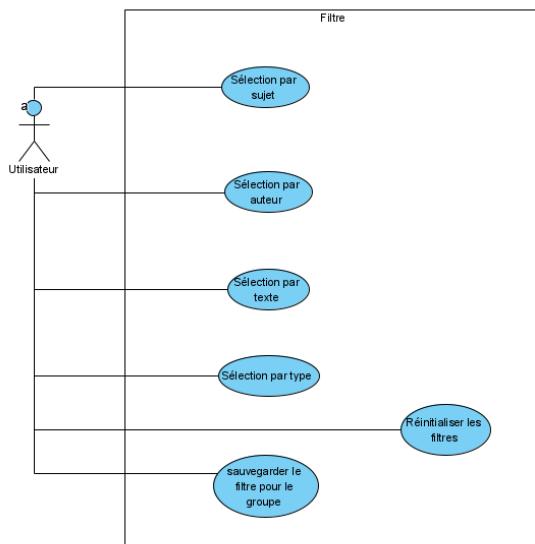
**FIG. 14 :** Use case : gestion des fils de discussion

**3.3.2.4 Notification** Les paramètres de notifications seront globaux et non spécifique à une tâche ou groupe



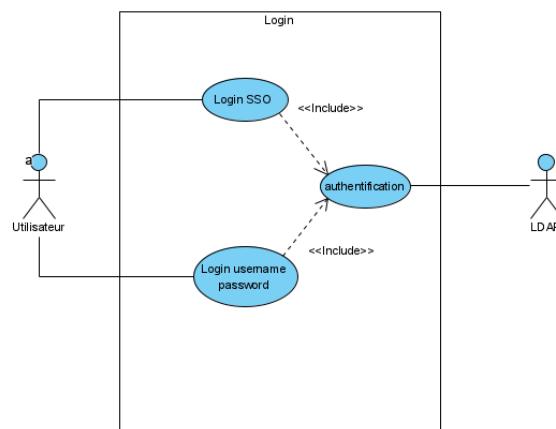
**FIG. 15 :** Use case : gestion des notifications

**3.3.2.5 Filtre** Use case concernant les différentes actions possibles avec les filtres.



**FIG. 16 :** Use case : gestion des filtres

**3.3.2.6 Login** Use case concernant les différentes actions possibles lors de l'authentification.



**FIG. 17 :** Use case : gestion du login

Les différents schémas ont été produits avec l'outil Visual Paradigm community edition<sup>5</sup>.

<sup>5</sup><https://www.visual-paradigm.com/>

### 3.3.3 Gestion des filtres

Les filtres permettent d'affiner la liste des tâches disponibles. Ils permettent de retrouver une tâche aisément et de retrouver une tâche passée ce qui n'est pas possible sur l'ancienne version de Caravel.

**3.3.3.1 Liste des filtres** Voici une liste exhaustives des filtres qui sont disponibles dans l'application :

- Par sujet
- Par auteur
- Par texte (texte entré par l'utilisateur, recherche dans le titre ainsi que la description)
- Par type de tâches (projet, devoir, CP/examen)
- Par état (clos/ouvert -> passé/futur)
- Par privé/publique

Il sera possible en tout temps de réinitialiser les filtres appliqués pour retrouver l'affichage standard.

### 3.3.4 Système de réactions

Le système de réactions sur les tâches a pour but de signaler la présence d'un problème sur ces dernières. L'idée derrière ces réactions est de donner une direction sur la réflexion à entreprendre pour évaluer la problématique d'une tâche. Dans cette optique les réactions ont pour but de juger de manière qualitative (en terme méthodologique) une tâche.

Pour ce faire nous allons définir une base de réactions qui sera la même sur toutes les tâches et que les élèves pourront utiliser. Il est important que cette base soit commune à toutes les tâches afin que les élèves puissent appréhender correctement leur utilisation. En effet des réactions spécifiques à chaque sujet ou groupe demanderait un effort d'assimilation trop conséquent et placerait l'élève dans une situation d'incertitude quant au choix de la réaction, ce qui serait contre productif. Le système doit rester simple et pouvoir être assimiler facilement.

La liste exhaustives des réactions :

-  Trop long
-  Trop complexe
  - manque de compétences
-  Manque d'informations

- donnée pas claire
-  Je suis perdu
  - la préparation en cours n'est pas optimale pour entreprendre l'exercice
-  Lien avec le cours pas clair
  - l'intérêt n'est pas clair, pas assez motivé, l'importance du devoir n'est pas comprise par l'étudiant
-  Peu d'intérêt
  - Par exemple pas de feedback, l'étudiant ne voit pas d'intérêt de s'investir

Plusieurs références ont été utilisées pour déterminer ces réactions :

- Un article de journal écrit dans le American Journal of Engineering Education (AJEE) (voir 2).
- Ainsi que deux autres articles en ligne voir références (3) et (4).

**3.3.4.1 Inciter à réagir** Afin de pousser les utilisateurs à réagir sur les différentes tâches, un système de déclenchement pourra être mis en place, en substance, il s'agit de regarder les tâches terminées dans un certains laps de temps très court (1-2 jours) selon un taux de probabilité défini : notifier l'utilisateur afin qu'il réagisse à une tâche, la réaction n'est pas obligatoire.

**3.3.4.2 Anonymisation** La question s'est posée quant à l'anonymisation des résultats, une réflexion a été portée en ce sens : le fait d'anonymiser les résultats n'apporte pas de désavantage tandis que l'inverse peut freiner les utilisateurs à donner leur avis. Le choix s'est donc porter sur des réactions anonymes.

### 3.3.5 Système de notifications

Les notifications sont des éléments importants car ils permettent aux utilisateurs de toujours rester à jour par rapport au contenu de leurs groupes.

**3.3.5.1 Canaux de distribution** Les différents canaux de distribution visés, sont :

- PWA / interne à l'application
- Email

**3.3.5.2 Déclenchement des notifications** La liste exhaustive des actions qui peuvent déclencher une notification :

- Ajout d'une tâche
- Modification d'une tâche
- Suppression d'une tâche
- Ajout d'une question dans une tâche
- Ajout d'un commentaire si abonné ou si auteur
  - (par défaut si un utilisateur répond à une question ou s'il est auteur, il devient automatiquement abonné)
- Demande d'ajout au groupe
- Accepté dans un groupe
- Refusé d'un groupe

Les différentes notifications peuvent être paramétrables depuis le compte de l'utilisateur.

	Mail	PWA/Interne
Trigger 1	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Trigger 2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Trigger 3	<input type="checkbox"/>	<input type="checkbox"/>
Trigger 4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**FIG. 18 :** Maquette des paramètres de notifications

La figure 18 donne une possible de représentation des paramètres.

### 3.3.6 Gestion de la charge de travail

Afin de mieux estimer la charge de travail, chaque branche accueillera un nouveau paramètre, le nombre de crédits ECTS<sup>6</sup> (un crédit représente une charge de travail d'environ 25 à 30 heures de travail). Ces crédits servent de critère de pondération pour les différentes indicateurs.

**3.3.6.1 Indicateurs** A partir de ces données une cotation **par semaine** est créée : le Work Load Score (WLS). Il s'agit d'un ratio basé sur un autre indicateur, le "Week Effort Score" (WES) et de sa médiane définie par les semaines déjà effectuées.

<sup>6</sup>[https://fr.wikipedia.org/wiki/Système\\_européen\\_de\\_transfert\\_et\\_d'accumulation\\_de\\_crédits](https://fr.wikipedia.org/wiki/Système_européen_de_transfert_et_d'accumulation_de_crédits)

Les détails des calculs sont donnés par les formules suivantes :

$$N_A = \text{Nombre de devoirs}$$

$$N_E = \text{Nombre d'Examens ou CP}$$

$$N_{PS} = \text{Nombre de projet en cours (qui ne sont pas à rendre)}$$

$$N_{PW} = \text{Nombre de projet à rendre}$$

$$C_S = \text{Nombre de crédits pour le sujet (cours)}$$

$$\text{Week Effort Score (WES)} = \sum_{\text{subjects}} C_s * (N_E + N_A + N_{PW} + 2 * N_{PS})$$

$$\text{Work Load Score (WLS)} = \frac{WES}{\widetilde{WES}}$$

**3.3.6.2 Comptabilisation des projets** Les projets sont calculés de manières différentes car on comptabilise un projet sur lequel on doit travailler mais qui n'est pas à rendre cette semaine et un projet qui est à rendre dans le courant de la semaine, ce qui engendre en général plus de travail.

**3.3.6.3 Normalisation** Le nombre de crédit n'est pas normalisé car tous les cours ne seront pas forcément présents en tout temps et donc il n'est pas possible d'avoir une normalisation homogène si des sujets viennent s'ajouter au fur et à mesure (dans le cas où ces informations ne sont pas calculées en temps réel).

**3.3.6.4 Gestion des extrema** Le score de certaines semaines risque de poser des problèmes, il faut donc éviter les extrema afin d'avoir une tendance qui soit plus cohérente. Pour éviter ça, la médiane des semaines est utilisée afin d'évaluer si une semaine est plus ou moins chargée

---

Projet N°227 : Caravel

He-Arc

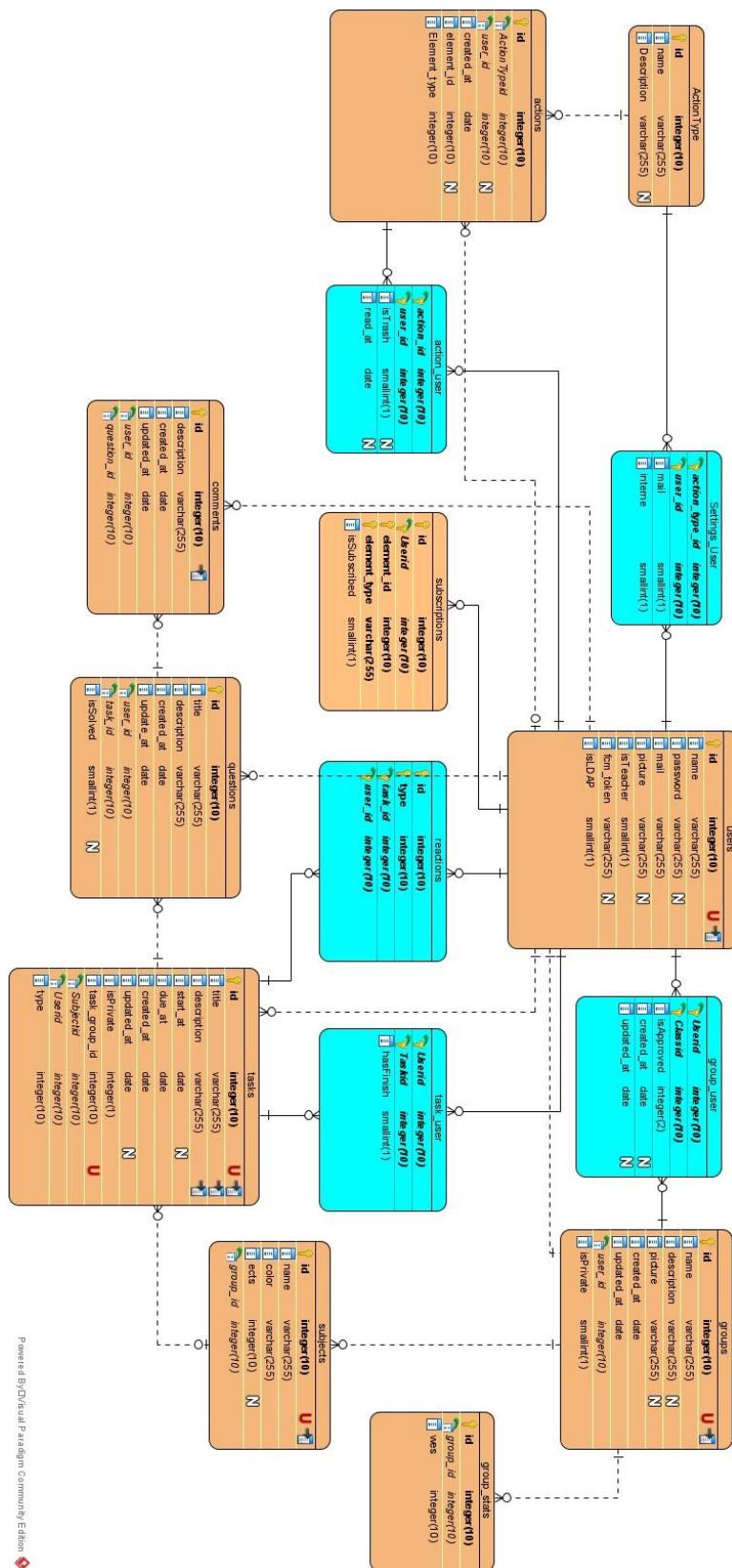
29 juillet 2021

### **3.3.7 Modélisation de la base de données**

La base de donnée a été modélisée avec l'outil Visual Paradigm<sup>7</sup>.

---

<sup>7</sup><https://www.visual-paradigm.com/>



**Fig. 19 :** Modélisation de la base de données

Dans la figure 19, les relations **Many-to-Many** sont exprimées en bleu.

### 3.3.8 Système d'authentification

L'actuel Caravel utilise un système d'authentification interne à l'application, il est donc nécessaire que chaque nouvel utilisateur s'enregistre avant de pouvoir se connecter sur l'application. Afin de faciliter cette démarche, une solution de SSO avec Google et GitHub a été mise en place. Dans la nouvelle version de Caravel, on souhaite permettre à l'utilisateur de se connecter via l'annuaire interne de l'école (LDAP). Cela permettra de directement récupérer des infos pertinentes sur l'utilisateur ainsi que de déterminer son rôle (professeur ou élève). L'utilisation du LDAP devrait permettre à terme de pouvoir enrôler les utilisateurs directement dans des classes.

## 3.4 Définitions des routes

Les routes sont définies en utilisant le principe REST et donc avec l'utilisation des verbes HTTP : GET, POST, PUT/PATCH, DELETE. La génération des routes est documentée avec l'outil en ligne Swagger (OpenAPI), sur lequel on peut retrouver la documentation de l'API Caravel<sup>8</sup>.

## 3.5 Stratégie & conception de test

Cette partie décrit la stratégie ainsi que la conception des tests nécessaires au bon fonctionnement de l'application. Il renseigne aussi les risques liés au projet.

### 3.5.1 Attentes de la qualité du produit

- Interface facile à utiliser
- Bonne qualité de code
- Site conforme aux normes standard du Web

### 3.5.2 Objectifs de tests

Le but des différents tests est de s'assurer que le code produit est de bonne qualité tant dans sa réalisation que dans son fonctionnement, en outre il permet de mettre en place des tests qui permettent de ne pas régresser d'une version à l'autre en maintenant une qualité de produit constante entre les différentes phases de développement.

<sup>8</sup><https://app.swaggerhub.com/apis-docs/M4n0x/Caravel/1.0.0#/>

Voici la liste des objectifs :

- Avoir un code maintenable
- Avoir un bon temps de réponse
- Permettre une charge d'au moins 40 personnes

### 3.5.3 Périmètre de tests

- Test unitaire avec PHPUnit (et Jest côté Vue.js)
- Test d'utilisabilité (Usability test)
- Qualité du code avec SonarCloud

### 3.5.4 Gestion des risques

La gestion des risques s'effectue par le calcul suivant :

$$C = P * I$$

Où C = Criticité, P = Probabilité [0,1] et I = impact [0,10].

Description	Source	Probabilité	Impact	Criticité	Résolution
Retard sur le planning	Interne	0.4	6	2.4	Découper les tâches de manière à facilement pouvoir évaluer le temps de mise en place (éviter les tâches avec trop d'action en même temps)
Login LDAP	Interne	0.7	5	3.5	Voir la documentation, rapidement voir avec un professeur, réévaluer la faisabilité

Projet N°227 : Caravel

He-Arc

29 juillet 2021

Description	Source	Probabilité	Impact	Criticité	Résolution
Mauvaise évaluation de la charge de travail du à l'absence de connaissance approfondie sur certaines technologies	interne	8	7	5.6	En référer le plus rapidement possible au mandant et adapter les objectifs en fonction de retard pris

### 3.5.4.1 Etapes principales

1. Tests unitaires PHPUnit (et Jest côté Vue.js)
2. Tests avec les utilisateurs
3. Analyser la qualité de code avec SonarCloud
4. Analyser les résultats dans le rapport de tests

### 3.5.4.2 Environnement et outils de tests

**3.5.4.2.1 GitHub** Pour l'intégration continue et la livraison continue GitHub sera utilisé.

**3.5.4.2.2 SonarCloud** La version cloud de SonarQube sera utilisée afin d'analyser la qualité du code.

## 3.6 Maquettes

Cette section regroupe les différentes maquettes créées pour la nouvelle version de Caravel. Ces maquettes ont été réalisées avec l'outil Figma<sup>9</sup> avec une licence étudiante.

### 3.6.1 Page de login

Cette maquette présente la page avec laquelle l'utilisateur pourra se connecter.

---

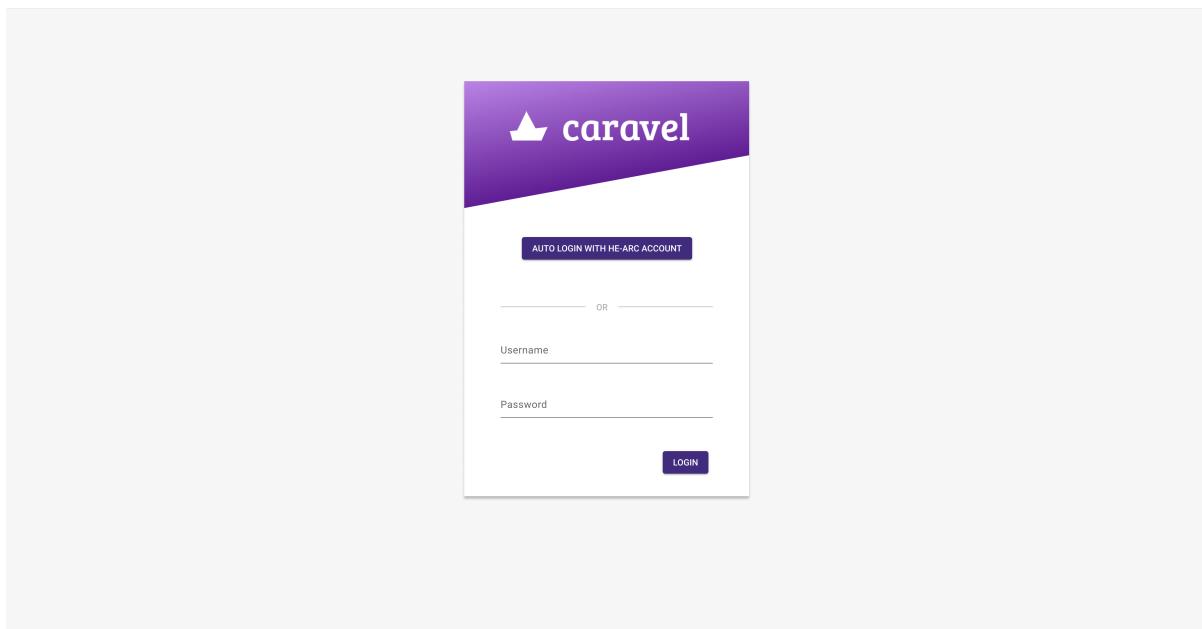
<sup>9</sup><https://www.figma.com/>

Projet N°227 : Caravel

He-Arc

29 juillet 2021

 caravel



**FIG. 20 :** Maquette : page de login

### 3.6.2 Page des tâches

Cette page a pour but d'afficher de manière chronologique les différentes tâches à faire, ainsi en un coup d'œil il est possible de voir quelles sont les tâches les plus prioritaires.

La maquette montre une interface web pour la gestion de projets et de tâches. En haut à gauche, il y a un logo "caravel". En haut à droite, il y a un menu avec "John Doe". Le menu principal comprend "INF DLM-B 2020", "Tâches (17)", "Calendrier", "Time line", "Statistiques" et "Paramètres".

Le contenu principal est divisé en deux sections :

- Recherche et filtres :** Un champ de recherche "Rechercher une tâche" avec un bouton "Nouvelle tâche". Des filtres peuvent être appliqués par "Sujet", "Auteur", "Type", "Etat" et "privé".
- Tâches :** Les tâches sont regroupées par date :
  - 2 Jours :** "Processus de normalisation" (statut "Ouvert"), "Test sur les ondes telluriques" (statut "Terminé")
  - 5 Jours :** "Processus de normalisation" (statut "Ouvert"), "Test sur les processus de normalisation" (statut "Ouvert"), "Processus agiles" (statut "Ouvert")
  - 31 Mars :** "Projet N°2 compilateur" (statut "Ouvert")
- Projets :** Une liste de projets avec leur statut :
  - "Projet N°2 compilateur" (statut "Statut 3")
  - "Projet Qt" (statut "Statut 1")
  - "Projet Unity" (statut "Statut 4")

En bas de la page, il y a une pagination avec les numéros 1 à 6.

**FIG. 21 :** Maquette : page liste des tâches

### 3.6.3 Page d'affichage d'une tâche

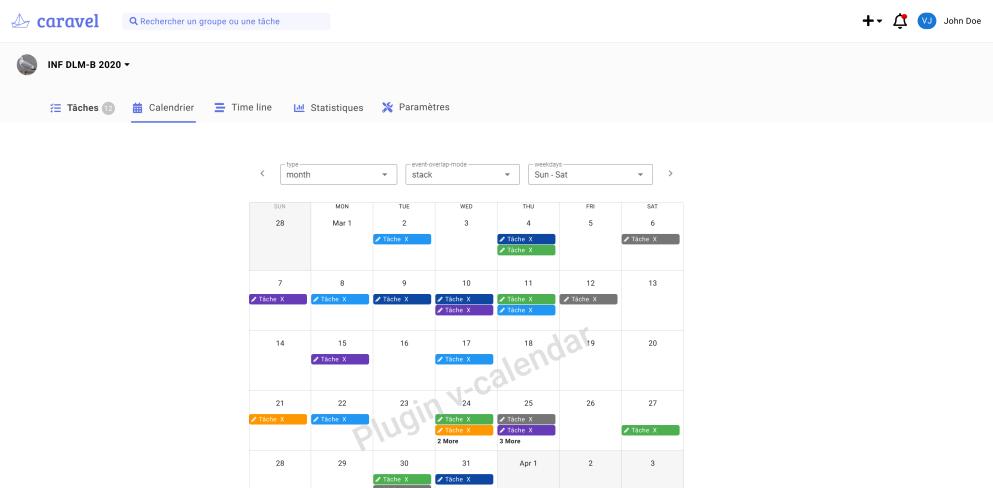
Cette maquette présente l'affichage d'une tâche et des éléments qui la composent.

La maquette montre une interface web pour la gestion de projets. En haut, un menu avec "caravel" et une barre de recherche. Ensuite, une section pour les groupes et tâches, où il est indiqué "INF DLM-B 2020". Les options "Tâches", "Calendrier", "Time line", "Statistiques" et "Paramètres" sont disponibles. Une tâche spécifique intitulée "Processus de normalisation" est affichée, ouverte par "John Doe" le 31 mars. La tâche contient du texte en français et des boutons pour "SUIVRE" et "MARQUER COMME FAIT". En dessous, une liste de questions avec leurs réponses et statut (en vert). Enfin, une section pour ajouter une nouvelle question avec champs pour le titre et le corps, et un bouton "AJOUTER".

**FIG. 22 :** Maquette : page affichage d'une tâche

### 3.6.4 Page vue mensuelle

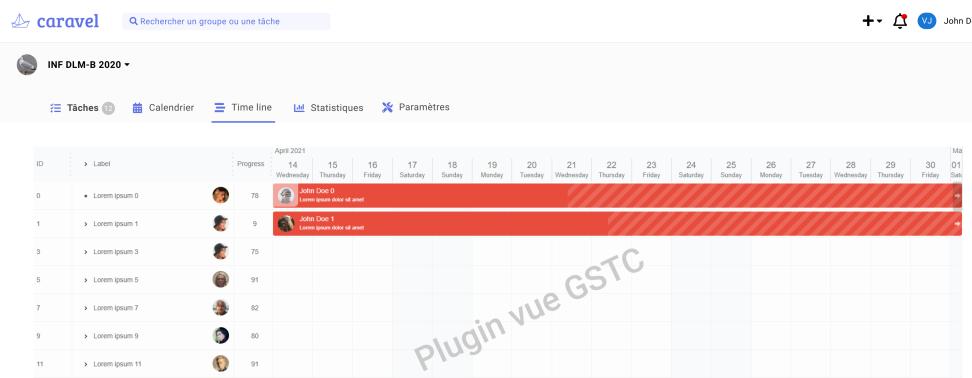
Cette maquette présente la vue mensuelle sur laquelle la charge de travail peut être facilement visible.



**FIG. 23 :** Maquette : page vue mensuelle

### 3.6.5 Page chronologique

Cette maquette représente la vue “Gantt” ou encore appelée vue “chronologique”, elle permet de voir l’ensemble des tâches futures.



**FIG. 24 :** Maquette : page vue timeline

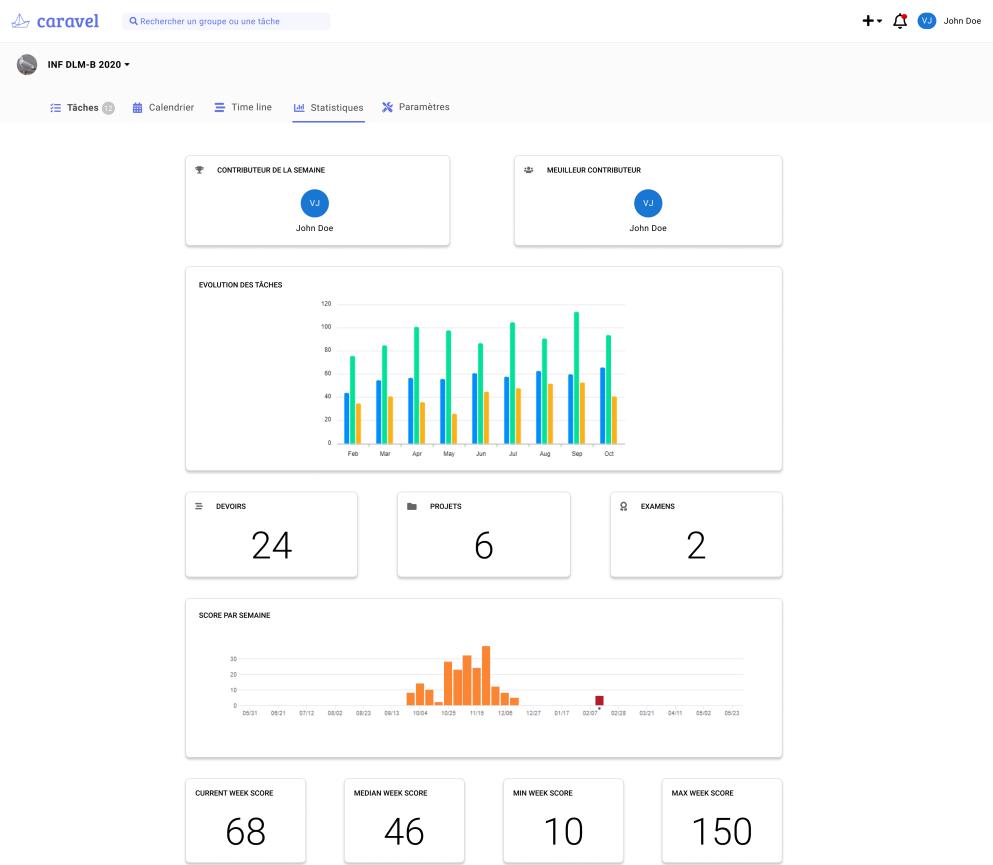
### 3.6.6 Page de statistiques

Cette maquette représente ce à quoi pourrait ressembler la page de statistiques,

Projet N°227 : Caravel

He-Arc

29 juillet 2021



**FIG. 25 :** Maquette : page de statistiques

### 3.6.7 Maquette interactive

Il est de plus possible de consulter la version interactive<sup>10</sup> de la maquette directement sur le site de Figma.

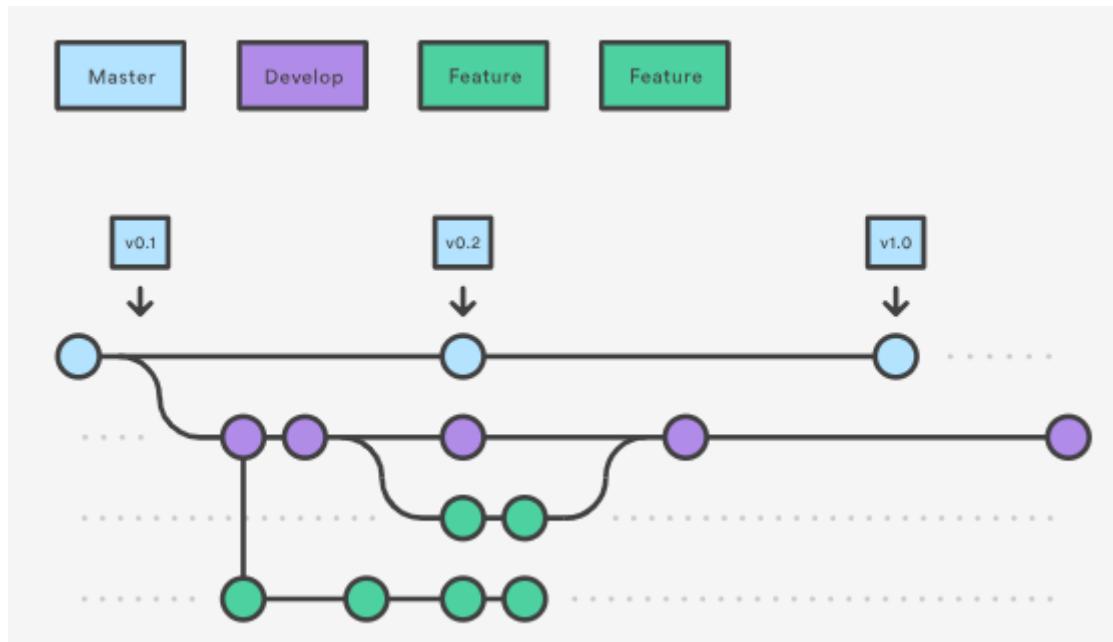
## 3.7 Planning

Voir annexe 9.2.

<sup>10</sup><https://www.figma.com/proto/WHGPKvp8GgmoqsaOP7mFlz/Caravel-mockup>

### 3.8 Méthodologie de travail

La méthodologie de travail se base sur l'utilisation GitFlow.



**FIG. 26 :** Méthodologie : GitFlow workflow feature branches

Cette méthodologie, représentée par la figure 26, consiste à créer 3 différentes canaux :

1. La branche `master` qui est une branche toujours fonctionnelle et stable (`release candidate`)
2. La branche `develop` qui possède les dernières fonctionnalités mais n'est pas forcément stable
3. Les branches dites `features` sont créées pour chaque nouvelle fonctionnalité.

Lorsqu'une `feature` est aboutie et est validée par l'équipe de développement, elle est alors poussée sur la branche `develop` pour validation, pour une fois que la branche `develop` est considérée comme stable, celle-ci peut être poussée sur la branche `master`.

Cette méthodologie implique une bonne analyse en amont des tâches à effectuer ainsi qu'un découpage minutieux des tâches afin de garder des branches `features` simple et concise. Cela permet un suivi clair de l'avancement du projet ainsi qu'une revue plus efficace de chaque nouvelle fonctionnalité mais demande un effort supplémentaire (création d'une branche et d'une pull request pour chaque fonctionnalité).

## 4 Implémentation

Dans cette section il s'agit d'expliquer les différentes étapes majeures qui ont permis la réalisation du projet ainsi que d'expliciter les différents choix techniques effectués.

### 4.1 DevOps CI/CD

Cette section explique comment a été mis en place le déploiement automatique ainsi que les différentes pipeline de test.

#### 4.1.1 Intégration continue

L'intégration continue consiste à faire des livraisons continues ainsi que de mettre en place des tests afin de vérifier que ces livraisons soient stables. Ces livraisons continues sont en partie réalisées grâce à la méthodologie GitFlow et aux GitHub actions

**4.1.1.1 Laravel** Laravel possède deux pipelines de test, une qui utilise MySQL (MariaDB) et une autre SQLite, cette façon de faire nous garantit une certaine abstraction entre l'utilisation de la base de donnée et notre code, en effet chaque moteur de base de données possède des particularités en utilisant deux systèmes de base de données on peut garantir l'interpolation entre les deux différents types de base de donnée.

En plus du test de connexion à la base de donnée, les tests unitaires PHPUnit sont lancés en fin de traitement pour les deux pipelines, comme nous pouvons le constater dans le code suivant :

```

1 # Fichier : .github\workflows\laravel.yml
2
3 name: Laravel CI SQLite fast
4
5 on:
6   push:
7
8 defaults:
9   run:
10    working-directory: ./backend
11
12 jobs:
13   tests:
14     runs-on: ubuntu-latest
15
16   steps:
17     - uses: actions/checkout@v2

```

```

18      - name: Copy .env
19        run: php -r "file_exists('.env') || copy('.env.example', '.env');"
20
21      - name: Setup PHP
22        uses: shivammathur/setup-php@v2
23        with:
24          php-version: '7.4'
25      - name: Install composer Dependencies
26        run: composer install -q --no-ansi --no-interaction --no-scripts
27          --no-progress --prefer-dist
28      - name: Generate key
29        run: php artisan key:generate
30      - name: Directory Permissions
31        run: chmod -R 777 storage bootstrap/cache
32      - name: Create Database
33        run: |
34          mkdir -p database
35          touch database/database.sqlite
36      - name: Execute tests (Unit and Feature tests) via PHPUnit
37        env:
38          DB_CONNECTION: sqlite
39          DB_DATABASE: database/database.sqlite
40        run: |
41          php artisan migrate --seed
42          vendor/bin/phpunit

```

**Code 1:** DevOps : pipeline de test Laravel SQLite

La version MySQL est similaire à cette dernière, seule la configuration de la base de données change.

#### 4.1.2 Livraison continue

La livraison est une étape qui consiste à déployer de manière automatique dès qu'une modification de code est effectuée. Ainsi notre application reflète toujours l'état actuel du code.

**4.1.2.1 Configuration du serveur** Le déploiement automatique ne s'occupe que de mettre les données de l'application à jour, elle ne s'occupera pas de la configuration totale du serveur qui nécessite plusieurs composants indépendants (NPM, Nginx, Composer, PHP, MariaDB, etc...). Il faut donc s'atteler à créer une configuration minimale du serveur pour accueillir notre backend ainsi que notre frontend.

```

1 #Server database mariadb
2 sudo apt install mariadb-server
3
4 # NPM pour VueJs
5 sudo apt install npm

```

```

6
7 # PHP et de ses dépendances
8 sudo apt install php7.4 libapache2-mod-php7.4 php7.4-curl php-pear php7
    .4-gd php7.4-dev php7.4-zip php7.4-mbstring php7.4-mysql php7.4-xml
    curl php7.4-ldap -y
9
10 # Composer pour les dépendances php
11 sudo apt install composer

```

**Code 2:** Serveur : installation des dépendances de base

Le code 2 installe les différentes composants nécessaires pour le lancement de Caravel.

**4.1.2.2 Configuration de MariaDB** Afin de garder la base de donnée de Caravel dans un vase clos, un utilisateur spécifique est créé pour accéder aux données de Caravel.

```

1 CREATE DATABASE Caravel;
2
3 grant all privileges on Caravel.* TO 'Caravel'@'localhost' identified
    by 'PLACERHOLDER_PASSWORD';
4
5 flush privileges;

```

**Code 3:** Serveur : création de la db et d'un user particulier

**4.1.2.3 Configuration de Nginx** Dans un premier temps, il a été choisi de séparer le backend et le frontend via deux sous domaines différents à savoir `caravel.ing.he-arc.ch` pour la partie frontend et `api.caravel.ing.he-arc.ch` pour la partie backend mais après plusieurs tests, il semble que cela ne soit pas possible en tout cas en l'état, car le sous domaine `api` n'est pas redirigé sur le serveur, il faudrait donc ajouter une entrée DNS supplémentaire au niveau du service informatique. Afin de ne pas perdre de temps sur la partie configuration, il a été choisi en définitive de faire passer l'api sur une route spécifique c'est-à-dire `caravel.ing.he-arc.ch/api`. La configuration suivante du Nginx reflète ce dernier choix.

```

1 # Caravel conf
2 server {
3     index index.php index.html;
4     root /var/www/caravel/backend/public;
5     server_name caravel.ing.he-arc.ch;
6     client_max_body_size 210M;
7
8     location / {
9         root /var/www/caravel/frontend/dist;
10        try_files $uri /index.html;
11    }

```

```

12
13     location /api {
14         try_files $uri $uri/ /index.php?$query_string;
15     }
16
17     location ^~ /uploads/ {
18         root /var/www/caravel/backend/public/;
19     }
20
21     location ~ \.php$ {
22         fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
23         fastcgi_index index.php;
24         fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
25         include fastcgi_params;
26     }
27
28     listen 443 ssl http2; # managed by Certbot
29     ssl_certificate /etc/letsencrypt/live/caravel.ing.he-arc.ch/
30         fullchain.pem; # managed by Certbot
31     ssl_certificate_key /etc/letsencrypt/live/caravel.ing.he-arc.ch/
32         privkey.pem; # managed by Certbot
33     include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
34         Certbot
35     ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
36 }
37
38 server {
39     if ($host = caravel.ing.he-arc.ch) {
40         return 301 https://$host$request_uri;
41     } # managed by Certbot
42
43     listen 80;
44     server_name caravel.ing.he-arc.ch;
45     return 404; # managed by Certbot
46 }
```

**Code 4:** Serveur : configuration Nginx

Dans le code 4, la partie SSL est directement gérée par CertBot<sup>11</sup> qui est un outil open source qui permet d'automatiquement enrouler les sites issues de la configuration Nginx avec des certificats Let's Encrypt<sup>12</sup> et donc d'activer l'HTTPS.

**4.1.2.4 Runners GitHub** Un runner GitHub est un agent qui s'installe sur un serveur distant, il permet de lancer directement des tâches sur ce dernier depuis des GitHub Actions<sup>13</sup>, cela est nécessaire

<sup>11</sup><https://certbot.eff.org/>

<sup>12</sup><https://letsencrypt.org/>

<sup>13</sup><https://github.com/features/actions>

pour effectuer le déploiement automatique. Pour les détails d'installation voir la page wiki dédiée<sup>14</sup> sur le GitHub de Caravel.

#### 4.1.3 Environnement de production

Certaines configurations sont dépendantes de l'environnement (base de données, LDAP, etc...), un fichier de configuration unique ne peut être défini pour tous les environnements. De même que ces derniers fichiers peuvent contenir des informations sensibles il est donc nécessaire de faire une configuration propre à chaque environnement dont elle seule détient les informations.

**4.1.3.1 Méthode par écrasement** La méthode pour gérer la configuration de la production consiste à écraser ou ajouter les fichiers nécessaires lors du déploiement automatique, pour ce faire la configuration est stockée sur un dossier spécifique `/var/www/config/caravel` et à chaque déploiement cette configuration est copiée dans le répertoire de déploiement. Il suffit donc de poser le fichier de configuration nécessaire que l'on veut surcharger dans le dossier en respectant la nomenclature du dossier cible, par exemple pour le fichier `auth.php` il faut donc déposer le fichier `auth.php` dans `/var/www/config/caravel/backend/config/` il sera alors automatiquement copié lors du déploiement.

**4.1.3.2 Dossier de téléchargement** Une des problématiques avec le déploiement c'est que la mise à jour passe par le remplacement de tous les fichiers de notre application par la dernière version disponible sur le repository GitHub, cependant les dossiers de téléchargement sont propres à chaque webapp, il ne faut donc pas les remplacer lors de la mise à jour du contenu de notre webapp.

Pour régler cette problématique, les deux dossiers de téléchargement :

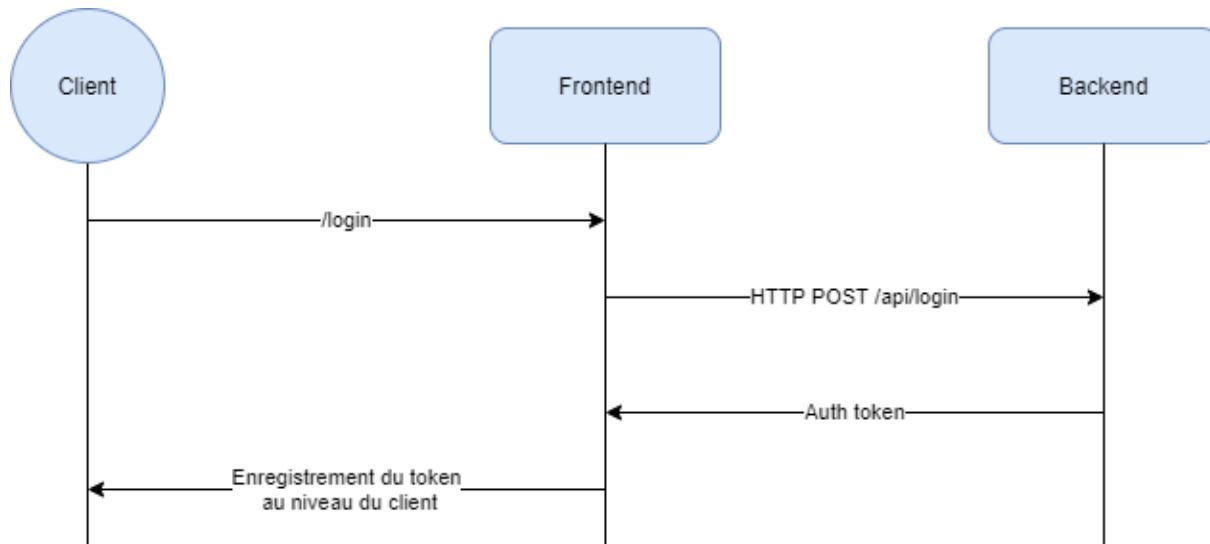
- `backend/storage`
- `backend/public/uploads`

sont, lors du déploiement, créés comme des liens symboliques sur des répertoires qui ne sont pas dans le répertoire de déploiement, ce qui évite que les dossiers ne soient à chaque fois écrasés par le déploiement d'une nouvelle version.

### 4.2 Authentification

Le processus d'authentification est un peu plus complexe dans une application où le frontend et le backend sont séparés, le processus peut être résumé simplement par le schéma suivant :

<sup>14</sup><https://github.com/HE-Arc/Caravel/wiki/CI-CD-Caravel-2.0>



**FIG. 27 :** Authentification : schéma d'interaction

Dans la figure 27, le processus apparaît de manière assez simple, dans un premier temps une requête est effectuée sur `api/login` afin d'authentifier l'utilisateur, si la validation est correcte alors un token est retourné à l'utilisateur, en fin ce token est enregistré au niveau du client pour authentifier les prochains appels à l'API.

#### 4.2.1 Local Storage vs Cookies

La complexité réside dans le choix du stockage du token au niveau du client, en effet une des solutions les plus utilisées est le stockage du token au niveau du Local Storage, cependant il s'agit d'une mauvaise pratique (voir référence 5).

L'autre solution consiste à utiliser les cookies ainsi que le flag `httpOnly` qui bloque l'accès javascript à ce dernier dès que ce flag est paramétré à `true` et c'est la solution qui est recommandée dans la documentation de Laravel, nous y reviendrons dans la section suivante.

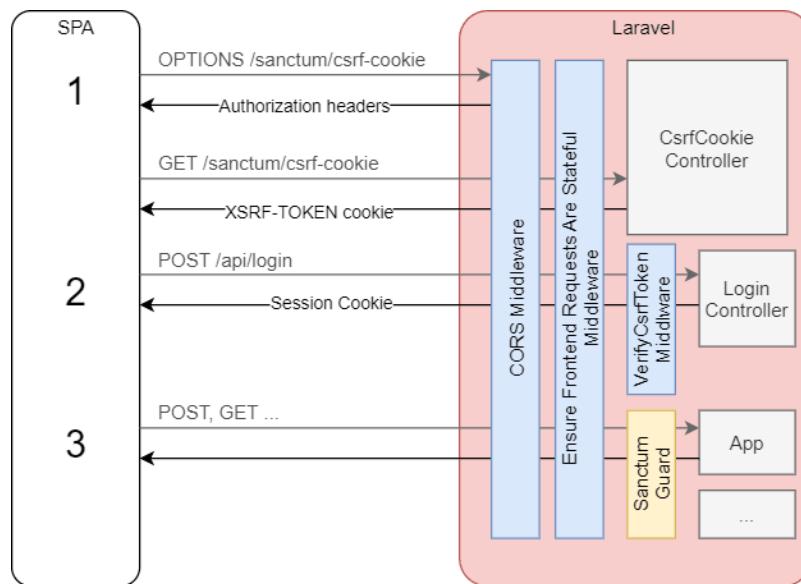
#### 4.2.2 Sanctum vs Passport

Laravel propose deux systèmes d'authentification, le premier Sanctum<sup>15</sup> est un système léger d'authentification basé sur des tokens, le second Passport<sup>16</sup> est un système d'authentification plus lourd qui utilise OAuth2. OAuth2 est un protocole qui permet aux utilisateurs la connection avec d'autres

<sup>15</sup><https://laravel.com/docs/8.x/sanctum>

<sup>16</sup><https://laravel.com/docs/8.x/passport>

applications externe tels que Google ou encore GitHub. Ce dernier est donc plus lourd et pré-suppose une bonne connaissance du protocole OAuth2. Comme l'utilisation de OAuth2 n'est pas nécessaire, Sanctum a été choisi, c'est d'ailleurs une recommandation issue de la documentation de Laravel<sup>17</sup>.



**FIG. 28 :** Laravel Sanctum Explained : SPA Authentication (voir 6 )

Dans la figure 28 on peut voir le fonctionnement de sanctum, les éléments les plus importants sont le CSRF token ainsi que le Session cookie. Le CSRF token permet de protéger l'utilisateur d'une cross-site request cet élément n'est pas en `httpOnly`. Le Session Cookie, qui est le fonctionnement normal que nous pourrions retrouver avec une session PHP, est le cookie qui garde les informations liés à l'utilisateur, cet élément est protégé par un `httpOnly` et n'est donc pas accessible via javascript. Ces deux paramètres de session sont placés par Laravel à l'appel de la route `/sanctum/csrf-cookie`. Il est donc important de faire un appel à cette route avant toute tentative de connexion.

```

1 //Fichier : frontend\src\store\modules\user.ts
2
3 await axios.get(process.env.VUE_APP_API_BASE_URL + "sanctum/csrf-cookie");
4
5 const response: AxiosResponse = await axios({
6   url: process.env.VUE_APP_API_BASE_URL + "login",
7   data: { mail, password },
8   method: "POST",
9 });

```

**Code 5:** Authentification : fonction de login

<sup>17</sup><https://laravel.com/docs/8.x/passport#passport-or-sanctum>

Le code 5 illustre la paragraphe précédent, nous pouvons voir qu'un appel à `sanctum/csrf-cookie` est effectué avant la tentative de connexion.

Nous pouvons constater en ligne 3, aucun retour particulier n'est attendu car Laravel va automatiquement inscrire les cookies nécessaires et la librairie utilisée pour les appels backend, Axios, va lui aussi de manière automatique faire les configurations nécessaires dès lors que le paramètre `axios.defaults.withCredentials = true` est positionné.

#### 4.2.3 Authentification LDAP

Une fois la configuration simple mise en place, il s'agit d'ajouter l'authentification LDAP. Dans le cas de la HE-Arc il s'agit d'une connexion à un Active Directory, pour effectuer des tests en local un OpenLDAP suffit.

```
1 docker run -d --rm -p 10389:10389 -p 10636:10636 rroemhild/test-openldap
```

**Code 6:** LDAP : création d'un annuaire avec docker

Ce docker permet de rapidement mettre en place un annuaire OpenLDAP, les informations sur le contenu (utilisateurs, groupes, OUs, etc...) se trouve sur le readme du GitHub<sup>18</sup>.

**4.2.3.1 LdapRecord vs Adldap2** Il existe actuellement deux outils pour effectuer la connexion à un LDAP, Adldap2<sup>19</sup> système de connexion LDAP éprouvé, bien documenté mais il n'est plus mis à jour à part pour la correction de bug. De l'autre côté, il existe LdapRecord<sup>20</sup>, vue comme le successeur d'Adldap2, il est facile d'utilisation et il existe une documentation spécifique pour Laravel.

Après discussion avec M. Nicolas Sommer et du fait que l'Adldap2 n'est plus maintenu que pour des bugfix, il a été choisi d'utiliser LdapRecord.

**4.2.3.1.1 Synchronisation LDAP** La connexion à un annuaire a pour but de simplifier la vie de l'utilisateur, ainsi nous pouvons récupérer des informations sur l'utilisateur sans l'intervention de celui-ci. Les champs synchronisés sont donnés dans le fichier `auth.php`.

```
1 // Fichier : backend\config\auth.php
2
3 'driver' => 'ldap',
4 'model' => LdapRecord\Models\ActiveDirectory\User::class, // Specify
Active Directory or OpenLDAP
```

<sup>18</sup><https://github.com/rroemhild/docker-test-openldap>

<sup>19</sup><https://adldap2.github.io/Adldap2>

<sup>20</sup><https://ldaprecord.com/docs/laravel/v2/>

```

5   'rules' => [],
6   'database' => [
7       'model' => App\Models\User::class,
8       'sync_attributes' => [ // champs synchronisés
9           'name' => 'cn',
10          'email' => 'mail',
11          'firstname' => 'givenName',
12          'lastname' => 'sn',
13          App\Ldap\AttributeHandler::class,
14      ],
15  ],

```

**Code 7:** LDAP : synchronisation des champs

Les champs synchronisés se trouvent sous le paramètre `sync_attributes`, à partir de là il peut s'agir d'une synchronisation un pour un, c'est-à-dire sans transformation, auquel cas il suffit de mettre juste le champ de la cible dans l'annuaire à synchroniser. Pour des champs plus complexe qui ne peuvent être simplement repris, dans lequel un traitement est nécessaire, il est possible de définir une classe pour gérer des cas spéciaux.

C'est justement ce qui est effectué pour la synchronisation du champ `isTeacher` car il n'existe pas de champ dans l'annuaire pour déterminer cette information directement, il faut donc la calculer à partir d'autres champs.

```

1 // Fichier : backend\app\Ldap\AttributeHandler.php
2
3 class AttributeHandler
4 {
5     public function handle(LdapUser $ldap, DatabaseUser $user)
6     {
7         //set isLdap, set isProf
8         $user->isLDAP = 1;
9         $allowedOUs = explode(";", env("LDAP_TEACHERS_OUs", ""));
10        $allowedOUs = array_filter($allowedOUs);
11
12        //check if user is in a allowedOU to
13        if (!empty($allowedOUs)) {
14            foreach ($allowedOUs as $dn) {
15                $ou = OrganizationalUnit::find($dn);
16                if ($ou && $ldap->isDescendantOf($ou)) {
17                    $user->isTeacher = 1;
18                    return;
19                }
20            }
21        }
22
23        $user->isTeacher = 0;
24    }
25 }

```

**Code 8:** LDAP : gestion de paramètre spécifique

C'est donc avec le code 8 que nous déterminons si l'utilisateur est un professeur ou non. Basiquement il s'agit de vérifier la présence de l'utilisateur dans certaines OUs qui sont déterminées par le paramètre `LDAP_TEACHERS_OUs` dans le fichier de configuration `backend\ .env`.

**4.2.3.1.2 Mécanisme de login** Une fois que la configuration LDAP est en place, il faut paramétrer le comportement du login LDAP, en effet quel champ `LdapRecord` doit-il vérifier dans l'annuaire, est-ce avec le mail, le nom d'utilisateur, etc...

Il faut donc enrichir la mécanique standard pour ajouter notre propre mécanique, cela se passe dans `AuthAPIController.php`.

```

1 // Fichier : backend\app\Http\Controllers\Auth\AuthAPIController.php
2
3 public function login(Request $request)
4 {
5     $credentials = $request->only('mail', 'password');
6     $credentials2 = [
7         'sAMAccountName' => $credentials['mail'],
8         'password' => $credentials['password'],
9         'fallback' => [
10             'email' => $request->mail,
11             'password' => $request->password,
12         ]
13     ];
14
15     if (Auth::attempt($credentials) || Auth::attempt($credentials2)) {
16         // auth successful
17         ...
18     } else {
19         // auth failed
20         ...
21     }
22 }
```

**Code 9:** LDAP : gestion du login et du fallback

Pour le projet Caravel, il a été décidé que l'utilisateur devrait pouvoir se connecter avec son compte mail, pour des raisons pratiques nous avons ajouté la possibilité de se connecter simplement avec son nom d'utilisateur. Depuis le code 9, c'est la partie `$credentials2`, de plus cette partie contient un `fallback`, ceci permet de se connecter avec un compte existant uniquement sur Caravel et n'ayant pas d'existence sur le LDAP.

## 4.3 Intégration des routes

Dans le cas d'un frontend et d'un backend séparés, le travail au niveau des routes doit se faire deux fois. Basiquement les routes liés au backend sont toutes préfixées par `/api/*`, toutes les autres routes concernent le frontend.

### 4.3.1 Routes backend

Les routes qui ont été réellement implémentées diffèrent quelque peu des routes qui ont été conceptualisées, les routes suivantes ont été ajoutées :

- `/api/profile/fcmToken`, [POST|DELETE]
- `/api/profile/markAsRead` [POST]
- `/api/groups/{group}/reactions` [POST]
- `/api/groups/{group}/finished` [POST]
- `/api/profile/notifications` [GET]

La nouvelle documentation des routes à jour est disponible sur [Swagger<sup>21</sup>](#).

### 4.3.2 Routes frontend

Les routes concernant le frontend ne sont pas très intéressantes à développer, elles concernent uniquement les URLs telles qu'elles apparaissent au niveau du navigateur de l'utilisateur. Elles peuvent être directement consultées depuis le fichier `frontend/src/router/index.ts`.

## 4.4 Gestion des réactions

Cette section décrit comment sont gérés les réactions sur les tâches.

### 4.4.1 Gestion de l'identifiant

Chaque réaction est liée par un type, un utilisateur et une tâche, ces 3 éléments constituent le clé composée de cet élément, pour des raisons de simplicité les réactions possèdent tout de même une notion d'identifiant (les clés composées n'étant pas gérées dans Laravel).

<sup>21</sup><https://app.swaggerhub.com/apis-docs/M4n0x/Caravel/FINAL>

#### 4.4.2 Représentation des réactions

Les réactions sont anonymes il n'est donc pas possible de transmettre les réactions comme tels sans quoi les identifiants des différents utilisateurs apparaîtraient, un attribut personnalisé a été créé dans le model `Task.php` afin d'anonymiser les différentes réponses au niveau de l'API.

```

1 // Fichier : backend\app\Models\Task.php
2
3 /**
4  * Get all reactions list with hasReact related to the current logged
5  * user
6 public function getReactionsListAttribute()
7 {
8     $data = [];
9     $react = [];
10    $reactionList = [];
11
12    foreach ($this->reactions as $reaction) {
13        if (!isset($data[$reaction->type])) $data[$reaction->type] =
14            [];
15        $data[$reaction->type][] = $reaction;
16        if ($reaction->user_id === Auth()->id()) $react[] = $reaction->
17            type;
18    }
19
20    foreach ($data as $key => $reactions) {
21        $reactionList[] = [
22            'type' => $key,
23            'count' => count($reactions),
24            'hasReact' => in_array($key, $react),
25        ];
26    }
27
28    return $reactionList;
29 }
```

**Code 10:** Gestion de la serialisation des réactions

Dans le code 10, nous pouvons voir comment sont sérialisés les réactions pour chaque tâche. L'identifiant de l'utilisateur n'est donc plus transmis cependant il faut être en mesure de savoir si l'utilisateur a effectué une des réactions, c'est exactement le rôle du champ `hasReact` qui indique si l'utilisateur courant a effectué la réaction, ce champ est propre à chaque utilisateur (requête).

#### 4.4.3 Affichage des réactions

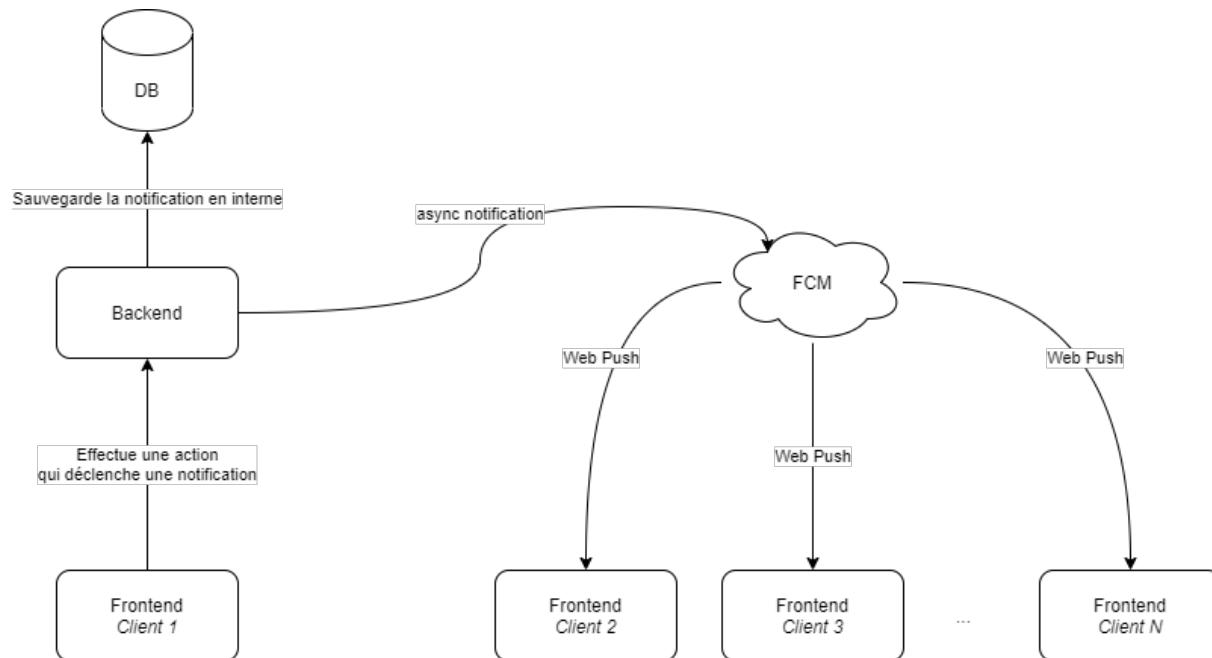
L'affichage des réactions est très simple, il est basé essentiellement sur le fichier de langue `frontend\src\locales\fr.json` et sur le fichier `frontend\src\components\task\Reactions.vue`. Il suffit donc simplement de rajouter ou de supprimer une réaction dans le fichier et d'adapter le nombre de réactions disponibles dans le fichier `Reactions.vue` pour changer les réactions qui seront affichées.

#### 4.4.4 Incitation aux réactions

Le système d'incitation aux réactions décrit dans la section 3.3.4.1 n'a pas pu être mis en place par manque de temps.

### 4.5 Système de notification

Le système de notification se divise en deux parties, la première qui est le déclencheur des notifications et la seconde qui consiste à récupérer les notifications. Ce découpage en deux parties est relative au découpage back et front end.



**FIG. 29 :** Notification : schéma global du système de notifications

Sur la figure 29 nous pouvons voir le schéma global des transactions effectuées lors d'une notification.

Le déclenchement se produit avec le client 1, puis au niveau du backend nous avons deux actions qui sont effectuées, une première va enregistrer la notification en DB, l'autre va s'occuper d'envoyer une notification au serveur Firebase Cloud Messaging. Et finalement les différentes notifications vont être descendues sur les différents client via un système de WebPush<sup>22</sup> mis en place grâce aux outils fournis par Firebase.

#### 4.5.1 Déclenchement d'une notification

Pour déclencher une notification au niveau du backend, il faut qu'une modification ait lieu sur une tâche, une question ou un commentaire. Pour détecter ces changements des observers<sup>23</sup> ont été mis en place au niveau du backend. Dès qu'une action parmi les types *Création*, *Mise à jour* et *Suppression* est effectuée, ces observers sont susceptible d'être appelés. Chaque observer a sa propre mécanique pour savoir quand il doit être déclenché et à qui les notifications doivent être envoyées.

**4.5.1.1 TaskObserver** Le TaskObserver est déclenché par les actions suivantes :

- Crédation
- Mise à jour
- Suppression

Lorsqu'une de ces actions est effectuée, le TaskObserver est alors appelé et les notifications sont envoyées directement à tous les membres du groupe auquel la tâche est rattachée.

**4.5.1.2 QuestionObserver** Le QuestionObserver est déclenché par les actions suivantes :

- Crédation
- Mise à jour

Lorsqu'une de ces actions est effectuée, le QuestionObserver est alors appelé et les notifications sont envoyées directement à tous les membres du groupe auquel la question est rattachée.

**4.5.1.3 CommentObserver** Le CommentObserver est déclenché par les actions suivantes :

- Crédation

Lorsqu'une de ces actions est effectuée, le CommentObserver est alors appelé et les notifications sont envoyées directement à tous les participants de la question.

<sup>22</sup><https://developers.google.com/web/fundamentals/push-notifications>

<sup>23</sup><https://laravel.com/docs/8.x/eloquent#observers>

**4.5.1.4 Gestion de la diffusion des notifications** Lorsqu'une notification est créée, elle représente une classe particulière, la classe `Action.php`, c'est elle qui va induire le comportement de la notification, c'est à dire comment elle va être distribuée ou stockée. Elle possède donc une méthode propre qui donne les différents canaux de diffusion de la notification.

```

1 // Fichier : backend\app\Notifications\Action.php
2
3 /**
4  * Get the notification's delivery channels.
5  *
6  * @param mixed $notifiable
7  * @return array
8 */
9 public function via($notifiable)
10 {
11     //faire le check pour le user (notifiable) si les paramètres sont
12     //ok
13     return ['database', FcmChannel::class];
14 }
```

**Code 11:** Notifications : canaux de diffusion

Dans le code 11 nous pouvons voir que notre action va être stockée en DB et diffusée via le Firebase Cloud Messaging. Pour le canal `database` il s'agit d'un canal disponible par défaut dans Laravel, en ce qui concerne le FCM channel il s'agit d'un ajout externe<sup>24</sup>.

Pour chaque canal il faut ensuite déterminer son comportement via les méthodes appropriées, via les méthodes décrites dans le code suivant.

```

1 public function toFcm($notifiable) // Pour Firebase Cloud Messaging
2 {
3     return ...;
4 }
5
6 public function toArray($notifiable) // Pour l'enregistrement en DB
7 {
8     return ...;
9 }
```

**Code 12:** Notification : comportement des canaux

**4.5.1.4.1 Envoi asynchrone des notifications** Si beaucoup de membres sont dans le groupe cette action peut prendre beaucoup de temps, il est nécessaire que cette tâche ne bloque pas la requête du client, il est possible de faire en sorte de mettre les notifications dans une queue qui sera alors

<sup>24</sup><https://laravel-notification-channels.com/fcm/>

exécuté dans un autre thread. Pour cela il suffit de rajouter le trait `Queueable` à notre classe ainsi que l'interface `ShouldQueue`

```

1 class ... implements ShouldQueue
2 {
3     use Queueable;
4
5     ...
6 }
7 }
```

**Code 13:** Notification : envoi asynchrone

A partir de là Laravel s'occupe seul de faire le travail en détectant automatiquement l'interface `ShouldQueue`.

#### 4.5.2 Configuration de FCM

Pour la configuration de FCM au niveau du backend la documentation officielle (voir référence 7) doit être suivie. En ce qui concerne la configuration au niveau du frontend il faut se référer aux documents utilisés dans le cadre de ce projet :

- How to add FCM to vue.js (voir référence 8).
- Intégration de Firebase Cloud Message avec Laravel et Vue.js (voir référence 9).
- Documentation officielle de Firebase Cloud Message (voir référence 10).

**4.5.2.1 Changements par rapport à la conception** Par rapport à la conception les déclencheurs suivants n'ont pas été traités :

- Demande d'ajout au groupe
- Accepté dans un groupe
- Refusé d'un groupe

par manque de temps.

#### 4.5.3 Récupération des notifications depuis le frontend

Pour l'envoi de notifications aux utilisateurs, le backend a besoin de connaître le token FCM de l'utilisateur, ce token ne peut être obtenu que par le client, comme la notification est lancée depuis le backend pour des raisons de sécurité il faut donc transmettre ce token du front au backend.

```

1 const fcmToken = await firebase
2                     .messaging()
3                     .getToken({ vapidKey: process.env.
4                         VUE_APP_FIREBASE_VAPID_KEY });
5 auth.addFcmToken(fcmToken);

```

**Code 14:** Notification : enregistrement du token FCM

Cette transaction se fait depuis la page de login, si l'utilisateur accepte les notifications push, alors l'enrôlement du token FCM est effectué. Une route au niveau de l'api `api/profile/fcmToken` est donc disponible pour enregistrer le token depuis le frontend.

**4.5.3.1 Callback FCM** Une fois que le token est enregistré, nous pouvons récupérer les appels asynchrones issues de la librairie FCM.

```

1 // Fichier : frontend\src\fcm.ts
2
3 fire.messaging().onMessage((payload) => {
4     userModule.loadNotifications();
5     //Create notification push
6 }

```

**Code 15:** Notification : call asynchrone des messages

Depuis ce message asynchrone dans le code 15 deux opérations sont effectuées, la première consiste à rafraîchir les notifications interne de l'application et l'autre opération consiste à afficher la notification push sur le système cible (Android, Windows, etc...).

## 4.6 Frontend

Cette section décrit les détails techniques importants concernant frontend avec l'utilisation de `Vue.js`.

### 4.6.1 Configuration Vue.js

La configuration initiale du projet est importante, car elle décrit les fonctions qui seront utilisées tout au long du projet.

```

Vue CLI v4.5.13
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, PWA, Router, Vuex, CSS Pre-processors, Linter, Unit
? Choose a version of Vue.js that you want to start the project with 2.x
? Use class-style component syntax? Yes
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? Yes
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Sass/SCSS (with dart-sass)
? Pick a linter / formatter config: Prettier
? Pick additional lint features: Lint on save
? Pick a unit testing solution: Jest
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) 
  
```

**FIG. 30 :** Frontend : configuration Vue.js

Dans la figure 30, il est possible de voir la configuration utilisée pour ce projet.

**4.6.1.1 Version 2 vs version 3** Au début du projet, la version 3 de Vue.js venait de faire son apparition, à cause de sa meilleure intégration de TypeScript par défaut, la question s'est posée de savoir s'il était plus pertinent de travailler avec la version 2 ou la version 3 de Vue.js.

L'avantage de la version 3 repose essentiellement sur l'intégration par défaut de TypeScript, cependant les changements opérés dans cette nouvelle version ont rendu caduc beaucoup de projets qui fonctionnaient sur la version 2 mais nécessitent une mise à jour pour la version 3 de Vue.js. Ceci inclut par exemple Vuex (que nous verrons plus en détails dans la section 4.6.2) qui n'était pas disponible en version stable en début de projet.

Le choix s'est donc porté sur la version 2 de Vue.js pour des raisons de stabilité avec une intégration "manuelle" de TypeScript.

**4.6.1.2 Utilisation de TypeScript** L'utilisation de TypeScript dans la version 2 de Vue.js a complexifié énormément le développement de Caravel car son intégration se base en partie sur l'utilisation de modules externes qui ne sont pas officiellement supportés, en voici une liste exhaustive.

- Vue-class-component<sup>25</sup>
- Vuex-Module-decorators<sup>26</sup>
- Vue-property-decorator<sup>27</sup>

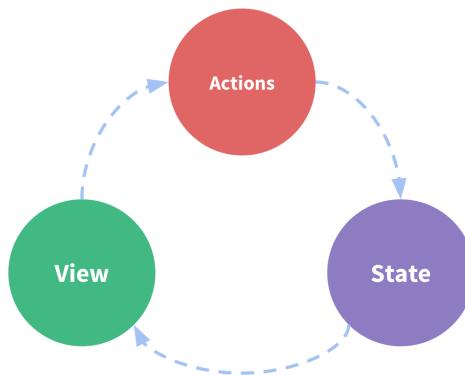
<sup>25</sup><https://class-component.vuejs.org/>

<sup>26</sup><https://github.com/championswimmer/vuex-module-decorators>

<sup>27</sup><https://github.com/kaorun343/vue-property-decorator>

#### 4.6.2 Vuex

Vuex est un gestionnaire d'états (state management pattern) pour Vue.js. Il est utilisé comme source centrale d'informations au sein de vue, avec des modificateurs d'état permettant de garantir qu'un état n'est modifié que de façon prédictible.

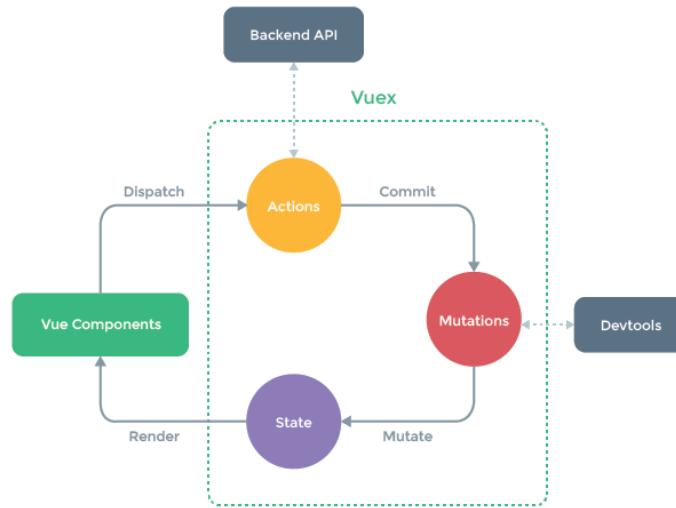


**FIG. 31 :** Vuex : one-way data flow, voir référence (11)

Dans la figure 31, nous pouvons voir le principe de fonctionnement global de Vuex, une vue déclenche une action, qui elle-même déclenche un changement d'état d'une donnée qui sera de nouveau utilisée par une vue.

Le principe devient utile lorsque plusieurs composants dépendent de la même source d'informations et que ceux-ci peuvent tous induire un changement sur le contenu. Sans Vuex la manière de modifier ce contenu peut être implémentée de manière très différente à travers les différents composants et rendre les changements incohérents. De plus il arrive souvent qu'un composant enfant nécessite les propriétés d'un de ses parents qui n'est pas direct, sans l'utilisation de Vuex il est nécessaire de faire descendre cette propriété à travers tous les composants intermédiaires avant que l'enfant puisse acquérir cette information (cette dernière problématique peut aussi être résolue avec l'utilisation des inject/provide<sup>28</sup>).

<sup>28</sup><https://vuejs.org/v2/api/#provide-inject>

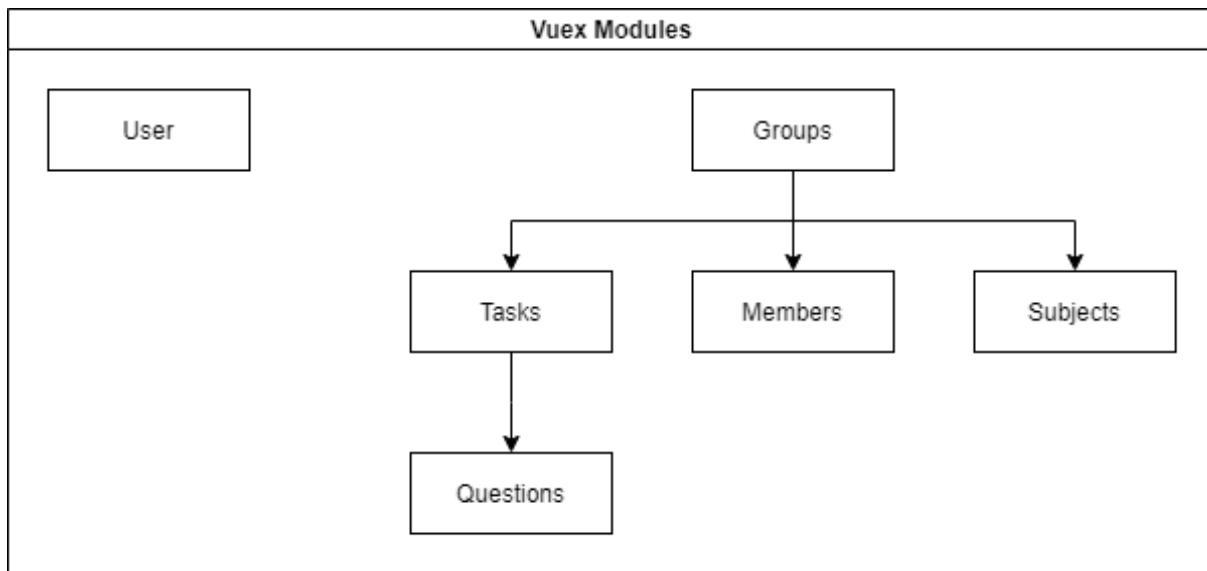


**FIG. 32 :** Vuex : détails de fonctionnement, voir référence (11)

Dans la figure 32, lorsqu'un composant veut faire une modification, il doit forcément faire appel à une action qui va s'occuper d'initier les mutations nécessaires sur l'état des éléments. Cet état est ensuite transmis dans les différents composants pour être utilisé/affiché.

Cette manière de fonctionner est un élément central de Caravel.

**4.6.2.1 Modules** Les modules permettent de récupérer ainsi que de faire des interactions sur des éléments spécifiques, voici une vue d'ensemble des différents modules ainsi que de leurs dépendances.



**FIG. 33 :** Vuex : liste des modules

Tous les modules peuvent communiquer entre eux, les liens de dépendances présents dans la figure 33 seront décrits dans les sections qui vont suivre.

De manière générale ces modules offrent des options CRUD dans leur domaine respectivement.

**4.6.2.1.1 User module** Le module `User` contient les informations sur l'utilisateur connecté, de plus il contient aussi toutes les méthodes de connexion et de déconnexion de l'utilisateur.

**4.6.2.1.2 Group module** Le module `Groups` permet de récupérer les groupes disponibles pour l'utilisateur courant. C'est le module principale de l'application car il est le chef d'orchestre des autres modules, tant que ce module n'est pas chargé avec l'action `loadGroup(id: string)` les données des modules "enfants" sont vides.

```

1  @Action
2  loadGroup(id: string): Promise<AxiosResponse> {
3      return new Promise((resolve, reject) => {
4          this.REQUEST();
5          axios({
6              url: process.env.VUE_APP_API_BASE_URL + `groups/${id}`,
7              method: "GET",
8          })
9          .then((response) => {
10             const group: GroupExtended = response.data;
11             TaskModule.load(group.tasks);
12             SubjectModule.load(group.subjects);
  
```

```

13     MemberModule.load(group.members);
14     this.LOADED();
15     resolve(response);
16   })
17   .catch((err) => {
18     this.ERROR();
19     reject(err);
20   });
21 });
22

```

**Code 16:** Vuex : chargement des dépendances

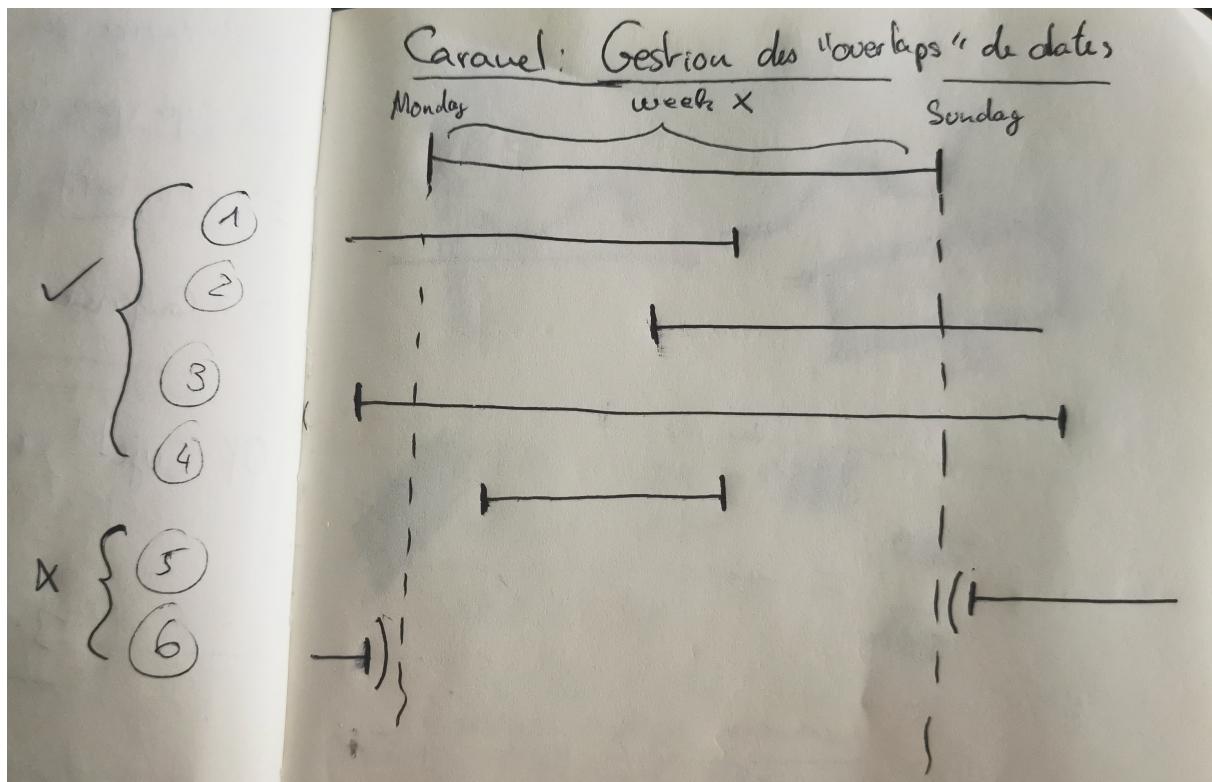
L'action `loadGroup`, qui est visible dans le code 16, a pour effet de charger les données pour des autres modules "enfants" tels que les modules `Task`, `Subject` et `Member`.

**4.6.2.2 Task module** Le module task est basé sur le même principe que le module de groupe, il possède la liste des tâches du groupe ainsi que la tâche sélectionnée si cela a lieu d'être.

**4.6.2.2.1 Calcul des statistiques** Le module `Task` détient une particularité supplémentaire, c'est le calcul des statistiques du groupe, en effet c'est dans ce module que les statistiques, c'est à dire le WES ainsi que le WLS, sont calculés.

Le choix du calcul au niveau du frontend à pour but de rendre les vues statistiques dynamiques, dès l'ajout de la moindre tâche ou changement de crédit au niveau des sujets, toutes les statistiques sont dynamiquement recalculées et mise à jour dans l'interface. La deuxième option qui consistait à faire les calculs au niveau du backend aurait demandé beaucoup plus de complexité. En effet si par exemple une nouvelle tâche est créée, alors il faut effectuer une seconde requête au backend pour récupérer les nouvelles statistiques. En utilisant l'avantage des propriétés réactives de Vue.js on s'affranchit de ces requêtes supplémentaires et de la création de route particulière au niveau du backend.

**Récupération des projets** Nous ne reviendrons pas sur les détails qui sont présents dans la section 3.3.6.1. Cependant une problématique qui peut être intéressante à soulever est la récupération des projets pour une semaine spécifique. En effet pour calculer la charge d'une semaine, il est nécessaire de récupérer les projets en cours.



**FIG. 34 :** Charge de travail : gestion des projets traversants

Par rapport à la figure 34, les cas qui nous intéressent sont les cas 1, 2, 3 et 4. Les cas 4 et 5 doivent être exclus. D'instinct, et sans schéma préalable, il serait tentant de produire un code pour gérer les cas 1 à 4 uniquement. Cependant chacun de ces cas requiert beaucoup de conditions. Prenons le cas 1, pour que le projet soit pris en compte dans une certaine semaine il faut que la date de fin du projet se situe après la date du début de la semaine **et** avant la fin de celle-ci. Gérer les cas 1 à 4 revient à gérer énormément de conditions.

Grâce à la représentation visuelle on peut voir la problématique dans l'autre sens. Au lieu de prendre les cas 1 à 4, nous pouvons récupérer tous les projets qui ne sont pas des cas 5 ou 6. Cela revient à faire uniquement deux conditions :

- Le début du projet ne se situe pas après la fin de la semaine
- La fin du projet ne se situe pas avant le début de la semaine

Ainsi récupérer les différents projets d'une semaine est beaucoup plus trivial.

**4.6.2.2 Modules : members, questions et subjects** Les modules `members`, `questions` ainsi que `subjects` sont des modules “standards”, ils n’ont pas de particularité spécifique et ne seront

donc pas détaillés dans ce document.

**4.6.2.2.3 Code冗長** En créant les différents modules, il s'est avéré que beaucoup de code se répétait, en profitant de typescript, une tentative de solution a été élaborée :

```

1 import { Data } from "@/types/data";
2 import { VuexModule, Mutation, Action } from "vuex-module-decorators";
3 import Vue from "vue";
4
5 // inheritance broken https://github.com/championswimmer/vuex-module-
6 // decorators/issues/125 wait for vue 3
7
8 export default abstract class DataModule<T extends Data> extends
9     VuexModule {
10     _items: T[] = [];
11     _status = "";
12     protected name = "";
13
14     get items(): T[] {
15         return this._items;
16     }
17
18     get status(): string {
19         return this._status;
20     }
21
22     @Mutation
23     protected ERROR(): void {
24         this._status = "error";
25     }
26
27     @Mutation
28     protected REQUEST(): void {
29         this._status = "loading";
30     }
31
32     @Mutation
33     protected FINISH(): void {
34         this._status = "loaded";
35     }
36
37     @Mutation
38     protected LOAD_ITEMS(items: T[]): void {
39         this._items = items;
40         this._status = "loaded";
41     }
42
43     @Mutation
44     protected UPSERT_ITEM(data: T): void {
45         const index = this._items.findIndex((item) => item.id == data.id);
46     }

```

```

44     if (index === -1) {
45         this._items.push(data);
46     } else {
47         Vue.set(this._items, index, data);
48     }
49 }
50
51 @Mutation
52 protected REMOVE_ITEM(data: T): void {
53     const index = this._items.findIndex((item) => item.id === data.id);
54     if (index !== -1) {
55         Vue.delete(this._items, index);
56     }
57 }
58
59 @Action
60 load(items: T[]): void {
61     this.LOAD_ITEMS(items);
62 }
63 }
```

**Code 17:** Vuex : module générique

Malheureusement l'utilisation du module vuex-module-decorators cause une erreur dans la gestion de l'héritage<sup>29</sup> de classe et il n'est donc pas possible d'utiliser cette version générique et elle est tout de même laissée à l'appréciation du lecteur à titre posthume et pour une utilisation future si le problème venait à être résolu.

**4.6.2.3 Gestion du chargement** La gestion du chargement est un élément important pour l'utilisateur, lorsqu'une action est effectuée un feedback doit être affiché à l'utilisateur afin d'éviter de la frustration et des comportements problématiques (plusieurs soumissions du même formulaire, etc...).

C'est ici que les status des modules sont intéressants, en effet lorsque le module effectue une requête, son status passe en mode "loading" lorsque que ce dernier a fini il passe sur le status "loaded". C'est ce que nous allons utiliser ici pour afficher un feedback lors du chargement.

```

1 // Fichier : frontend\src\views\GroupContainer.vue
2
3 get isGroupLoaded(): boolean {
4     return groupModule.status === "loaded";
5 }
6
7 get isTasksLoaded(): boolean {
8     return taskModule.status === "loaded";
9 }
10
```

<sup>29</sup><https://github.com/championswimmer/vuex-module-decorators/issues/125>

```

11  get isLoading(): boolean {
12    return this.isGroupLoaded && this.isTasksLoaded;
13 }
```

**Code 18:** Vuex : gestion du chargement

Le composant `GroupContainer` possède la propriété `isLoading` qui lui permet de savoir si les différents modules sont chargés, si ce n'est pas le cas il affiche une image de chargement.

Ceci est un exemple d'utilisation des status des modules, d'autres pages utilisent ce système ou des systèmes internes (comme le chargement sur le bouton dans les formulaires  )

#### 4.6.3 Gestion des erreurs Axios

La gestion des erreurs Axios peut être faite de manière unitaire, c'est-à-dire, chaque composant qui effectue une requête s'occupe seul de gérer toutes les erreurs issues d'une requête. Or si une partie des erreurs doit être gérée par le composant lui-même, il y a certaines erreurs qui devraient être gérées de manière globale et c'est justement une chose qui peut-être mise en place avec Axios en utilisant les interceptors.

```

1 Axios.interceptors.response.use(
2   (reponse) => reponse,
3   (error: AxiosError) => {
4     switch (error.response?.status) {
5       case 401:
6         if (router.currentRoute.name != "Login") {
7           userModule.logout();
8           router.push({
9             name: "Login",
10            query: { redirect: router.currentRoute fullPath },
11          });
12        }
13        break;
14      case 403:
15        if (router.currentRoute.name != "Forbidden") {
16          router.replace({ name: "Forbidden" });
17        }
18      }
19      return Promise.reject(error);
20    }
21  );
```

**Code 19:** Axios : gestion des erreurs globales avec les interceptors

Avec les interceptors, utilisé dans le code 19, les erreurs 401 (Unauthorized) ainsi que les erreurs 403 (Forbidden) sont gérées de manière automatique pour tous les appels effectués avec Axios.

#### 4.6.4 Vue router

Vue router est un module qui permet de gérer les différentes routes d'accès à Caravel. Il est utilisé pour gérer la navigation entre les différentes pages.

**4.6.4.1 Protection des routes** Certaines routes n'ont pas de raison d'être accédées lorsque l'utilisateur n'est pas connecté, il faut donc prévenir cette situation afin de rediriger l'utilisateur sur la page de login lorsque celui-ci tente d'accéder à une page nécessitant un utilisateur connecté.

Cette action est réalisée simplement avec Vue Router, en effet, il est possible de mettre en place des Navigation Guards<sup>30</sup>.

```

1 // Fichier : frontend\src\router\index.ts
2 router.beforeEach((to: Route, from: Route, next: NavigationGuardNext)
3   => {
4     if (
5       !auth.isLoggedIn &&
6       (to.meta.isAuthNeeded == undefined || to.meta.isAuthNeeded)
7     ) {
8       next({
9         path: "/login",
10        query: {
11          redirect: to.fullPath,
12        },
13      });
14    } else {
15      next();
16    }
17  });

```

**Code 20:** Vue.js : protection des routes

Avec le code ci-dessus, les routes sont par défaut protégées. Si une route n'a pas besoin d'être protégée, il suffit de rajouter la meta data `isAuthNeeded: false` sur la route en question, comme c'est le cas pour la route login qui doit être accessible sans authentification.

**4.6.4.2 Lazy loading routes** La taille d'une SPA (Single Page Application) peut devenir assez lourde au fur et à mesure des développements. Afin d'éviter d'avoir trop de données à charger lors de la première connexion, il est possible de différer le chargement des composants. Pour cela il suffit simplement de remplacer un import dans les routes par une fonction lambda, de cette manière le Webpack va automatiquement faire du code-split<sup>31</sup> et différer le chargement lorsque c'est nécessaire.

<sup>30</sup><https://router.vuejs.org/guide/advanced/navigation-guards.html>

<sup>31</sup><https://webpack.js.org/guides/code-splitting/>

```

1 // Fichier : frontend\src\router\index.ts
2 {
3   path: "/groups",
4   name: "GroupSearch",
5   component: () => import("../views/GroupSearch.vue"),
6 }

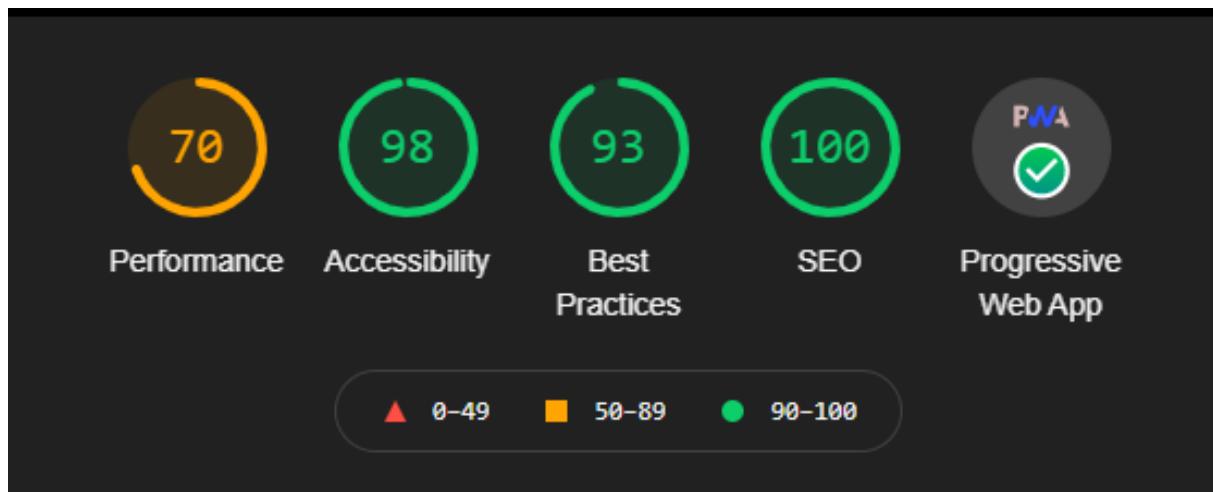
```

**Code 21:** Vue.js : exemple de lazy loading

Sur le code ci-dessus, le component `GroupSearch.vue` ne sera chargée que lorsque nous rentrons dans la route `/groups`. Cette technique est particulièrement intéressante car elle permet d'accélérer le premier chargement.

#### 4.6.5 PWA

La mise en place d'une application PWA est assez simple avec Vue.js, du moment que l'option a été spécifiée lors de la création du projet, il est simplement nécessaire que l'accès à l'application se fasse en HTTPS et de définir les icônes pour les différents types de mobiles ce qui peut être simplement fait avec les outils CertBot<sup>32</sup> et vue-pwa-asset-generator<sup>33</sup>.



**FIG. 35 :** PWA : score lighthouse

Sur la figure 35, il est possible de voir le résultat du score Lighthouse<sup>34</sup>.

<sup>32</sup><https://certbot.eff.org/>

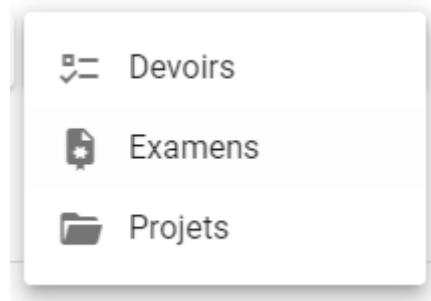
<sup>33</sup><https://github.com/jcalixte/vue-pwa-asset-generator>

<sup>34</sup><https://developers.google.com/web/tools/lighthouse>

#### 4.6.6 Composants

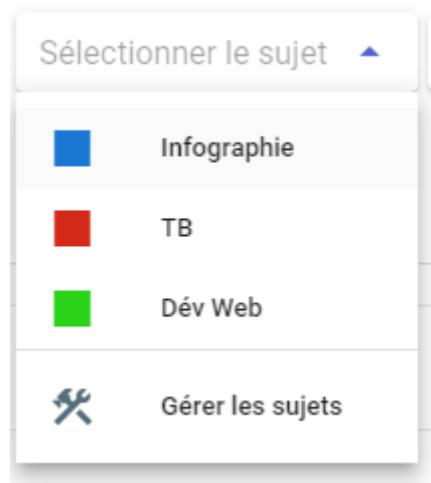
Plusieurs composants ont été développés dans le cadre du projet, cette section décrit quelques composants remarquables.

**4.6.6.1 Input spécifiques** Plusieurs inputs spécifiques ont été développé afin d'être utilisés à plusieurs endroits de l'application, c'est notamment le cas des inputs : `type` et `sujet` qui sont présents dans le filtre de recherche d'une tâche ainsi que dans le formulaire d'une tâche.



**FIG. 36 :** Vue.js : composant sélection du type

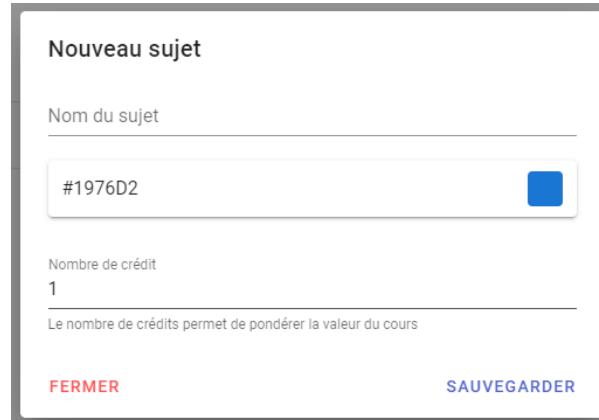
Sur la figure 36, il est possible de voir le rendu de l'input `type` dans le filtre de recherche.



**FIG. 37 :** Vue.js : composant sélection du sujet

Sur la figure 37, il est possible de voir le rendu de l'input `sujet` dans le filtre de recherche.

Le composant pour le sélectionner le sujet est un peu plus complexe que les autres composants car il possède la possibilité de créer directement un nouveau sujet. Grâce au système de composants, il a été possible de réutiliser la modale de création d'un sujet qui a été développée dans le cadre des paramètres du groupe :



**FIG. 38 :** Vue.js : modale de création d'une fenêtre

Sur la figure 38, il s'agit de la représentation de la modale sujet.

Ce composant est donc présent directement depuis la sélection des sujets mais est aussi présent dans la page des paramètres du groupe dans l'onglet sujets via les boutons Ajouter et Modifier.

**4.6.6.2 Pagination générique** La pagination est élément assez redondant dans l'application, elle nécessite souvent les mêmes paramètres en utilisant la force de Vue.js il est facile de créer un composant générique afin de respecter le principe DRY.

Un composant `Paginate.vue` a donc été créé en alliant le système de composant et le système de slot<sup>35</sup>, ce composant peut être utilisé comme ce suit :

```

1 <paginate :items="groups" :perPage="5">
2   <template #default="{ items }">
3     <group-item
4       v-for="group in items"
5         :group="group"
6         :key="group.id"
7         :hasJoin="false"
8         class="mt-3"
9       />
10    </template>
11  </paginate>
```

<sup>35</sup><https://vuejs.org/v2/guide/components-slots.html>

**Code 22:** Vue.js : exemple d'utilisation du composant paginate

Le principe est assez simple, l'élément pagination prend en entrée une liste d'éléments qu'il faut paginer, ici une liste de groupes, en interne il effectue le système de pagination nécessaire et ré-émet une liste des éléments visibles par rapport à la pagination actuelle, il suffit donc simplement d'expliciter comment ces éléments vont s'afficher dans le corps de l'élément `paginate` et le travail de pagination est terminé.

Pour voir la simplicité de la pagination il suffit de voir le code nécessaire à l'affichage de notre exemple sans la partie pagination :

```

1 <group-item
2   v-for="group in groups"
3   :group="group"
4   :key="group.id"
5   :hasJoin="false"
6   class="mt-3"
7 />

```

**Code 23:** Vue.js : code requis sans l'utilisation de la pagination

Les éléments nécessaires afin de créer une pagination sont donc très simples et basiques.

Le composant `paginate` s'occupe lui même d'afficher la pagination via l'utilisation de Vuetify<sup>36</sup> à la fin de la liste des éléments.

#### 4.6.7 Editeur markdown

L'éditeur markdown était un élément déjà présent dans l'ancienne version de Caravel, avec l'ajout du frontend Vue.js il a fallut changer d'éditeur, heureusement un éditeur similaire pour Vue.js est disponible : mavonEditor<sup>37</sup>.

Cette éditeur, cependant, n'est pas prévu pour l'utilisation de TypeScript ce qui a engendré pas mal de problématiques lors de son implémentation.

**4.6.7.1 Gestion des uploads** De base l'éditeur markdown mavonEditor ne permet que la gestion de téléchargement d'images, le composant a donc été modifié pour laisser la possibilité à l'utilisateur de pouvoir ajouter des fichiers de tous les types depuis le markdown. De plus il est maintenant possible de directement coller une image ou un fichier qui se trouve dans le presse-papier avec le raccourci `CTRL+V`.

<sup>36</sup><https://vuetifyjs.com/en/>

<sup>37</sup><https://github.com/hinesboy/mavonEditor>

## 4.7 Backend

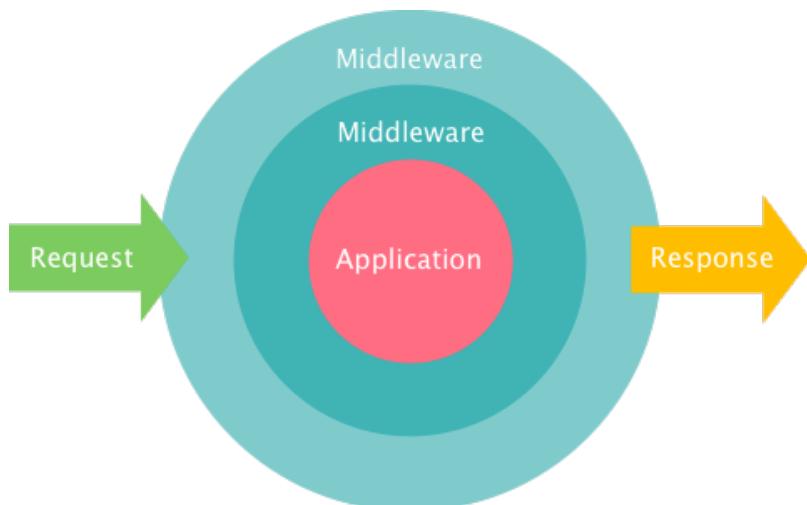
Cette section décrit les éléments importants dans l'utilisation de Laravel.

### 4.7.1 Choix de la base de données

Pour le choix de la base de données, il y a globalement deux possibilités qui s'imposent : PostgreSQL ou MariaDB (MySQL). Un article (voir référence 12) compare ces deux versions en terme de performances, il s'avère que MariaDB est plus performante sur des tables qui possèdent beaucoup de lignes (1 million pour le test effectué dans l'article) que PostgreSQL. Le choix s'est donc porté sur l'utilisation de MariaDB car à terme le nombre de tâches pourraient atteindre ce nombre de ligne.

### 4.7.2 Middleware

Un middleware est un composant de Laravel qui permet de d'introduire des mécaniques en entrée (agir sur la requête reçue par le serveur) ou en sortie (agir sur la réponse renvoyée par le serveur) au niveau de l'API.



**FIG. 39 :** Laravel : schéma de fonctionnement d'un middleware

Sur la figure 39 (source<sup>38</sup>) nous pouvons voir le fonctionnement global d'un middleware, il peut agir à l'entrée d'une requête ou à la sortie d'une réponse, dans notre cas c'est l'entrée qui va nous être utile. En effet pour certaines routes nous avons besoin de deux choses :

- Vérifier que l'utilisateur est bien authentifié

<sup>38</sup><https://blog.maqe.com/dealing-with-spaces-in-form-inputs-using-middleware-in-laravel-5-4-ffb37cd019e6>

- Vérifier que l'utilisateur a bien accès au groupe (si cela a lieu d'être)

Ces deux cas peuvent facilement être gérés avec des middlewares. Pour le login Sanctum fournit un middleware "out of the box" [App\Http\Middleware\Authenticate:sanctum](#). En ce qui concerne l'accès au groupe, un middleware a été créé, [CheckGroup](#), celui-ci vérifie de manière basique si un utilisateur a accès à un groupe ou non.

```

1 // Fichier : backend\app\Http\Middleware\CheckGroup.php
2 class CheckGroup
3 {
4     /**
5      * Handle an incoming request.
6      *
7      * @param \Illuminate\Http\Request $request
8      * @param \Closure $next
9      * @return mixed
10     */
11    public function handle(Request $request, Closure $next)
12    {
13        $group = $request->route('group');
14        $user = auth()->user();
15
16        if (empty($group)) {
17            return $next($request);
18        }
19
20        if (! $group instanceof Group) {
21            $group = Group::find($group);
22        }
23
24        $group->loadMissing('usersAccepted');
25
26        if (empty($group) || ! $group->usersAccepted->contains($user)) {
27            abort(403, __("api.global.403"));
28        }
29
30        return $next($request);
31    }
32 }
```

**Code 24:** Laravel : vérification des droits de groupe

Le code est donc assez simple, il ne s'agit que de vérifier que l'utilisateur se trouve dans les membres acceptés du groupe.

#### 4.7.3 Policies

En plus de l'accès aux groupes, il est possible de plus finement gérer les permissions sur les différentes modifications faites sur un élément dans Laravel grâce aux [Policies](#).

C'est ce qui a été utilisé pour les modifications faites aux groupes, cela permet de décharger le contrôleur et donc de mieux cloisonner les responsabilités.

#### 4.7.4 Validation des requêtes

Par rapport à la précédente version de Caravel, la vérification des paramètres était effectuée dans les contrôleurs, ce qui avait pour conséquences d'alourdir et de mélanger les différentes responsabilités au sein d'une même classe. Avec la nouvelle version, et sous les conseils de M. Visinand, les validations de requêtes ont été placées dans des classes dédiées.

```

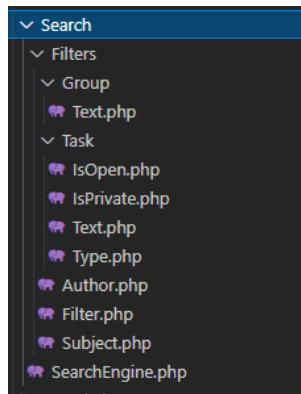
1 // Fichier : backend\app\Http\Controllers\CommentController.php
2 /**
3  * Create a comment
4  *
5  */
6 public function comment(CommentRequest $request, Group $group, Task
7     $task)
8 {
9     $comment = Comment::create($request->validated());
10    $comment->user_id = $this->user->id;
11    $comment->save();
12
13    return response()->json($comment);
14 }
```

**Code 25:** Laravel : exemple de contrôleur avec une classe de validation

Nous pouvons constater que pour le code ci-dessus, qui concerne la création d'un nouveau commentaire, le code est devenu très léger dans le contrôleur.

#### 4.7.5 Moteur de recherche

Afin de simplifier et de rendre plus flexible les différentes recherches, un pseudo moteur de recherche a été créé, basé sur l'article “Writing advanced Eloquent search query filters” (voir référence 13).



**FIG. 40 :** Laravel : structure des fichiers du moteur de recherche

Dans la figure 40, il est possible de voir la structure des fichiers mise en place pour la recherche sur Caravel.

Par rapport à l'article, un système de filtre spécifique a été ajouté. Dans l'article la recherche concerne uniquement un type, ici il est plus flexible nous pouvons spécifiquement ajouter un filtre pour un type. Ce dernier est particulièrement utile car la recherche est utilisée pour filtrer des tâches ainsi que pour rechercher de groupes.

```

1 // Fichier : backend\app\Http\Controllers\TaskController.php
2
3 /**
4  * List of tasks.
5  *
6  * @param Request $request
7  * @param Group $group
8  * @return \Illuminate\Http\Response
9 */
10 public function index(Request $request, Group $group)
11 {
12     $filters = $request->all();
13
14     $query = SearchEngine::applyFilters($group->tasks()->getQuery(),
15                                         $filters, "Task");
16
17     $tasks = $query->get();
18
19     return $tasks;
}
  
```

**Code 26:** Laravel : exemple d'utilisation du moteur dans un contrôleur

Le code 26 est donc beaucoup plus léger car toute la vraie complexité est cloisonnée dans les différents filtres.

**4.7.5.1 Filtre** Un filtre doit correspondre à un paramètre `GET` de l'url pour être activé, ici le principe de `convention over configuration` a été utilisé, ainsi pour que le filtre soit appelé le nom du paramètre `GET` doit avoir une nomenclature particulière.

Par exemple, pour l'url `/groups/1/tasks?isOpen=1`, le moteur va chercher un filtre avec le nom `IsOpen` (la première lettre est une majuscule) d'abord dans le folder `Search\Filters\[Model]\*`. Si l'il n'existe pas de filtre spécifique, le moteur va essayer de chercher un filtre global dans le folder `Search\Filters\*`. Si aucun filtre n'existe alors le paramètre sera simplement ignoré.

L'ajout d'un nouveau filtre est donc très simple, il suffit d'ajouter une nouvelle classe avec le nom voulu dans le dossier `Filters` si on souhaite un filtre global ou dans le dossier qui concerne le modèle si l'on souhaite un filtre spécifique.

```

1 class Text implements Filter
2 // Fichier : backend\app\Http\Search\Filters\Task\Text.php
3 {
4     /**
5      * Apply a given search value to the builder instance.
6      *
7      * @param Builder $builder
8      * @param mixed $value
9      * @return Builder $builder
10     */
11    public static function apply(Builder $builder, $value)
12    {
13        $text = "%$value%";
14        return $builder->where(function ($query) use ($text) {
15            $query->where('title', 'LIKE', $text)
16                ->orWhere('description', 'LIKE', $text);
17        });
18    }
19 }
```

**Code 27:** Laravel : exemple de filtre

La classe doit implémenter l'interface `Filter`, comme nous pouvons le voir dans le code 27, pour pouvoir être prise en compte.

## 5 Tests et validation

Cette section décrit les différents tests qui ont été mis en place pour vérifier la qualité de l'application.

## 5.1 Test unitaire

Les tests unitaires n'ont pas pu être réalisés, cependant ils ont été inclus dans la pipeline de validation ainsi l'ajout de tests unitaires au niveau de Laravel avec [PHPUnit](#) ou au niveau de [Jest](#) avec Vue.js sont automatiquement validés dans la pipeline DevOps.

## 5.2 Qualité du code

La mise en place de la qualité de code avec SonarCloud n'a pas pu être mise en place dans le temps imparti.

## 5.3 Test d'utilisabilité

Afin de tester globalement l'application, la réalisation d'un test utilisateur a eu lieu à St-Imier, le 07 juillet 2021.

Il y a eu au total 6 personnes interrogées dans le cadre de ce user test, les résultats des différents retours par les utilisateurs sont décrits dans le paragraphe qui suit, il s'agit essentiellement de données brutes.

### 5.3.1 Scénario de test

1. Connectez-vous sur Caravel
2. Vous êtes dans la classe "INF DLM-B 2021" et vous souhaiteriez créer un espace pour votre classe sur Caravel
3. Vous vous souvenez d'un devoir pour le mardi 20 juillet pour le cours d'infographie : "Lire tout le livre de WebGL" que vous souhaiteriez partager à votre classe à travers Caravel.
4. Vous prenez connaissance de l'existence du groupe "INF DLM-B 2019" et vous décidez de rejoindre le groupe.
5. Un utilisateur vient d'accepter votre demande d'accès à la classe "INF DLM-B 2019", vous souhaitez maintenant accéder au groupe pour voir ce qu'il contient.
6. En arrivant sur le groupe "INF DLM-B 2019" vous apercevez la tâche "Faire l'exercice 1" d'infographie, vous prenez le temps de le lire et décidez que ce travail n'est pas de votre niveau, vous souhaitez réagir à la tâche pour montrer votre opinion.
7. Malgré votre réaction, vous entreprenez quand même de réaliser le devoir, vous bloquez immédiatement sur le point 1a, vous souhaitez demander "Comment résoudre l'exo 1a" à vos camarades qui se trouvent sur le groupe.

8. En parcourant les diverses tâches du groupe “INF DLM-B 2019” vous apercevez la tâche “Séance de travail”, sur celle-ci se trouve une question “A quelle heure à lieu la séance?” Vous connaissez la réponse (17h30) et décidez d’y répondre.
9. En revenant sur votre question que vous avez posée “Comment résoudre l’exo 1a” vous apercevez que quelqu’un a répondu à votre question, la réponse vous convient, vous décidez que cette réponse est suffisante et passez l’état de la question en résolu.
10. Grâce à l’aide fournie par vos camarades sur la tâche “Faire l’exercice 1” d’infographie, vous avez réussi l’exercice, vous décidez de marquer cette exercice comme terminé pour vous.
11. Vous réalisez que la tâche que vous avez ajoutée “Lire tout le livre de WebGL” était une erreur vous décidez de supprimer cette tâche.
12. D’ailleurs vous décidez que le groupe “INF DLM-B 2019” est beaucoup mieux que le groupe que vous avez créé, comme ce dernier n’est plus utile et qu’il n’y a que vous, vous décidez alors de supprimer le groupe.
13. Finalement vous avez décidé de changer de classe, vous préférez donc quitter le groupe “INF DLM-B 2019”.

### 5.3.2 Résultats

Les différents résultats sont disponibles sur le wiki de Caravel<sup>39</sup>.

<sup>39</sup><https://github.com/HE-Arc/Caravel/wiki/R%C3%A9sultats-du-users-test-7-juillet>

## 6 Améliorations

Cette section décrit les possibles améliorations qui pourraient être apportées à Caravel.

### 6.1 Système de notification

Le système de notification est actuellement très basique, il ne contient pas d'options, ainsi l'utilisateur ne peut pas décider quel type de notification il souhaite recevoir. Un ajout majeure serait donc la possibilité de paramétrier depuis la page profile de l'utilisateur quel type de notification ce dernier souhaite recevoir.

#### 6.1.1 Système d'abonnement

Avec ceci il serait intéressant d'avoir un système d'abonnement sur les différents types, par exemple lorsqu'un utilisateur répond à une question il serait automatiquement "abonné" à la question, un bouton serait alors disponible sur le contenu en question pour lui permettre de se désabonner de ce dernier et donc de ne plus recevoir de notification liées.

### 6.2 Gestion des filtres en frontend

Actuellement les filtres sont gérés au niveau backend, l'idée initiale était de laisser la possibilité de faire une recherche transversale (à travers tous les groupes), cependant cette fonctionnalité n'est pas présente sur Caravel. La recherche d'une tâche se limite à un groupe. Dès lors effectuer une recherche en frontend serait plus judicieux car lors de la sélection d'un groupe toutes les tâches doivent être chargées car elles sont nécessaires dans certaines vues, comme les données sont donc déjà présentes un filtre en frontend serait plus rapide et plus adapté.

### 6.3 Editeur markdown

L'éditeur markdown est issu d'un plugin vue.js : mavonEditor<sup>40</sup>. Cet éditeur, bien que flexible, est très limité sur les fonctionnalités disponibles de plus il ne permet pas de faire uniquement du rendu, il faut impérativement rendre le composant en entier pour charger toutes les dépendances externes ce qui rend le composant très lourd pour uniquement faire de l'affichage. Une solution rapide qui est actuellement utilisée repose sur deux composants différents : un pour l'affichage et un pour l'édition du markdown. Les deux sont basés sur le plugin markdown-it<sup>41</sup>. Cependant qu'ils utilisent les deux

<sup>40</sup><https://github.com/hinesboy/mavonEditor>

<sup>41</sup><https://github.com/markdown-it/markdown-it>

markdown-it comme moteur de rendu, les options qu'ils utilisent dans markdown-it diffèrent légèrement, ce qui a pour cause de créer des différences entre l'affichage et l'édition.

Il faudrait donc une solution pour uniformiser le rendu dans l'éditeur markdown et celui dans le rendu final d'une tâche.

### 6.3.1 Ajout de balises spécifiques

Une fonctionnalité intéressante dans l'éditeur markdown serait celle de pouvoir ajouter des mentions à d'autre contenu comme par exemple le fait de mentionner un autre membre avec un @Member ou encore mentionner une autre tâche avec un #123 à la manière de ce que permet déjà GitHub.

## 6.4 Ajout des groupes de manière automatique

La création de groupe est uniquement manuelle, il serait intéressant de développer une fonctionnalité qui permette de créer automatiquement une liste des groupes avec des membres basé sur des groupes AD ou bien sur un simple fichier plat.

## 6.5 Ajout de paramètres de synchronisation avec LDAP

La synchronisation LDAP offre plusieurs avantages qui ne sont pas utilisés actuellement, typiquement la photo de l'utilisateur est disponible dans l'Active Directory mais n'est pas utilisée. Il serait intéressant de regarder s'il n'y a pas d'autres champs disponibles qui pourraient être utiles.

Actuellement les paramètres de l'utilisateur sont figés par les informations disponibles sur le LDAP, avec l'ajout de la photo, il faudrait laisser la possibilité à l'utilisateur de gérer ses paramètres de manière manuelle.

## 6.6 Gestion des suppressions

Actuellement Caravel ne s'occupe pas de nettoyer toutes les fichiers liés lorsqu'une tâche est supprimée. En effet tous les fichiers liés à la tâche sont conservés dans le dossier du groupe dans le [storage](#). La question est délicate car lorsqu'un fichier est uploadé au sein d'une tâche, cela crée une url au niveau du markdown pour ce fichier précis, ce lien peut alors être utilisé dans d'autres points de l'application, il faut donc faire en sorte d'être sûr que le fichier n'est plus mentionné dans aucun message avant de le supprimer sans quoi des "liens morts" pourraient apparaître.

## 7 Conclusion

Par rapport au premier objectif qui concernait le placement des devoirs et une meilleure visibilité de la charge de travail, l'implémentation du calcul de charge ainsi que l'affichage de ce dernier dans les différentes vues comme la vue chronologique ou encore la vue mensuelle, rendent possible une meilleure gestion du placement des devoirs et donc devrait permettre d'avoir une charge plus homogène.

Concernant le second objectif, qui est de réunir toutes les informations relatives au travail qui doit être effectué, cela est rendu possible par la création de tâches, ainsi que par la possibilité de charger des fichiers dans l'application mais aussi par la contribution collaborative sur ces tâches.

Le troisième objectif, qui est d'aider les élèves à mieux s'organiser, il est amplement rempli par les différentes vues qui permettent d'avoir un affichage clair des tâches à faire, de plus l'ajout du système de notifications permet d'être constamment à jour sur ces dernières.

Enfin le dernier objectif, qui était de proposer une plateforme de collaboration entre élèves et étudiants, l'objectif a été remplis grâce à l'implémentation d'un système de réactions ainsi que la possibilité de créer des questions dans les différentes tâches.

Nous avons donc une application fonctionnelle qui permet une meilleure gestion des devoirs au sein de l'He-Arc mais aussi une meilleure collaboration entre élèves et professeurs.

Pour conclure, l'application est loin d'être parfaite, il reste des points d'améliorations notables, notamment une meilleure gestion des notifications mais dans l'ensemble le projet Caravel a su répondre aux des objectifs principaux requis.

## 8 Glossaire

**SPA** Single Page Application

**PWA** Progressive Web App

**WLS** Work Load Score

**WES** Week Effort Score

**LDAP** Lightweight Directory Access Protocol

**DRY** Don't Repeat yourself

**CRUD** Create Read Update Delete

Projet N°227 : Caravel

He-Arc

29 juillet 2021

## 9 Annexes

**9.1 Guide d'installation**

**9.2 Planning (au format GanttProject et PNG)**

**9.3 Journal de travail**

## 10 Références

---

1. WIKIPÉDIA. *Méthode de MoSCoW* [en ligne]. 15 février 2021. [Consulté le 6 mars 2021]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/M%C3%A9thode\\_MoSCoW](https://fr.wikipedia.org/wiki/M%C3%A9thode_MoSCoW)
2. WENSHU, Li, RICHARD, Bennett, TAIMI, Olsen et RACHEL, McCord Ellestad. Engage Engineering Students In Homework : Attribution Of Low Completion And Suggestions For Interventions. [en ligne]. juin 2018. DOI 10.19030/ajee.v9i1.10186<sup>42</sup>. Disponible à l'adresse : [https://www.researchgate.net/publication/326191774\\_Engage\\_Engineering\\_Students\\_In\\_Homework\\_Attribution\\_Of\\_Low\\_Completion\\_And\\_Suggestions\\_For\\_Interventions](https://www.researchgate.net/publication/326191774_Engage_Engineering_Students_In_Homework_Attribution_Of_Low_Completion_And_Suggestions_For_Interventions)
3. DR. DAVIS BOWMAN, Jennifer. *Why Students Don't Do Their Homework-And What You Can Do About It* [en ligne]. 2016. [Consulté le mars 2021]. Disponible à l'adresse : <https://www.teachthought.com/pedagogy/why-students-dont-do-their-homework-and-what-you-can-do-about-it/>
4. DUTTA, Roger. *Six reasons why students fail to complete their homework* [en ligne]. 2020-04-01. [Consulté le mars 2021]. Disponible à l'adresse : <https://www.academicgates.com/blog/six-reasons-why-students-fail-to-complete-their-homework/43/view>
5. DEGGES, Randall. *Please Stop Using Local Storage* [en ligne]. 2018. Disponible à l'adresse : [htps://dev.to/rdegges/please-stop-using-local-storage-1i04](https://dev.to/rdegges/please-stop-using-local-storage-1i04)
6. BAILLY, Nicolas. *Laravel Sanctum Explained : SPA Authentication* [en ligne]. 2020. [Consulté le 20 juillet 2021]. Disponible à l'adresse : <https://dev.to/nicolus/laravel-sanctum-explained-spa-authentication-45g1>
7. EL-DIN, Mohamed Alaa. *Laravel FCM (Firebase Cloud Messaging) Notification Channel* [en ligne]. 2020-03-04. Disponible à l'adresse : <https://laravel-notification-channels.com/fcm>
8. TAF, Fima. *How to add FCM (Firebase Cloud Messaging) to VueJS* [en ligne]. 2021-03-08. [Consulté le 20 avril 2021]. Disponible à l'adresse : <https://dev.to/vbanditv/how-to-add-fcm-firebase-cloud-messaging-to-vuejs-37np>
9. CHRISTIANDY, Eddy. *Simple Laravel + Vue + Laravel Mix + Firebase Notification (PWA, Offline)* [en ligne]. 2019. [Consulté le 20 avril 2021]. Disponible à l'adresse : <https://gist.github.com/ehr/a1195e083e6ff4f980698fa06ebfec3>
10. GOOGLE. *Firebase Cloud Messaging* [en ligne]. 2021. Disponible à l'adresse : <https://firebase.google.com/docs/cloud-messaging>

---

Projet N°227 : Caravel

He-Arc

29 juillet 2021

11. VUE.JS. *What is Vuex?* [en ligne]. 2017. Disponible à l'adresse : <https://vuex.vuejs.org/>
12. KATHURIA, Pulkit. *Mysql vs Postgresql performance test with Laravel API for simple Eloquent queries on 1 million items* [en ligne]. 2019. [Consulté le juillet 2021]. Disponible à l'adresse : <https://medium.com/web-developer/mysql-vs-postgresql-performance-test-with-laravel-api-for-simple-eloquent-queries-on-1-million-6e0e6f1005b8>
13. CHOCHAN, Amo. *Writing advanced Eloquent search query filters* [en ligne]. 2016. [Consulté le juin 2021]. Disponible à l'adresse : <https://m.dotdev.co/writing-advanced-eloquent-search-query-filters-de8b6c2598db>